VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Realizace fázového závěsu pro standardní kartu sběru dat

Realization of phase lock loop for standard data acquisition board

2014 Lukáš Šoustek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Lukáš Šoustek**

Studijní program: B2649 Elektrotechnika

Studijní obor: 2601R004 Měřicí a řídicí technika

Téma: Realizace fázového závěsu pro standardní kartu sběru dat
Realization of Phase Lock Loop for Standard Data Acquisition Board

Zásady pro vypracování:

V měřicích systémech se jako jedna z metod dodržení konstantního poměru mezi frekvencí měřeného signálu a vzorkovací frekvence využívá princip tzv."fázového závěsu". Pro účely fázového závěsu HW cestou se využívají speciální integrované obvody. Při měření pomocí standardní měřicí karty pro PC není obvyklé, že je takový HW k dispozici. Náplní práce je nalézt metodu jak fázový závěs realizovat v LabVIEW.

Body zadání:
1. Seznámení se s funkcí fázového závěsu.
2. Seznámení se s funkcemi knihovny DAQmx pro měřicí karty.
3. Návrh a realizace fázového závěsu.
4. Ověření a rozbor funkce.
5. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] VLACH, Jaroslav, Josef HAVLÍČEK a Martin VLACH. *Začínáme s LabVIEW.* 1. vyd. Ilustrace Viktorie Vlachová. Praha: BEN - technická literatura, 2008, 247 s. ISBN 978-80-7300-245-9.
[2] BITTER, Rick, Taqi MOHIUDDIN a Matt NAWROCKI. *LabView advanced programming techniques.* 2nd ed. Boca Raton: CRC Press, c2007, 499 s. ISBN 08-493-3325-3.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Petr Bilík, Ph.D.**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Šoustek

Lukáš Šoustek

Poděkování

Rád bych poděkoval především svému vedoucímu bakalářské práce, panu doc. Ing. Petru Bilíkovi, Ph.D, za trpělivost, ochotu a pomoc při realizaci této práce.

Abstrakt

V kapitolách této práce se zabývám vývojem programu fázového závěsu (PLL), popisem vlastností každé úpravy a v závěrečné části testováním tohoto programu.

Program vyvíjím v prostředí LabVIEW, bude sloužit pro rozšíření možností využití běžných měřících karet a to z důvodu absence HW fázového závěsu na obvyklých kartách sběru dat.

Fázový závěs je zpětnovazební regulační obvod generující výstupní signál o stejné frekvenci a fázi (případně násobku frekvence) jako signál vstupní. V práci využívám principy PLL a to generování výstupní frekvence odpovídající násobku frekvence vstupního signálu.

Princip PLL využiji pro optimalizaci vzorkování neznámého periodického vstupního signálu tj. dodržování celistvého počtu period měřeného signálu v naměřených datech, pro další zpracování digitalizovaného signálu například aplikací FFT.

Klíčová slova

Frekvence, Fáze, Tw (časové okno), PLL, FFT, Měřící karta, DAQ, vzorkovací frekvence, LabVIEW

Abstract

In chapters of this work I have to develop program of phase loop lock (PLL), describes of any change what I make and in the end section testing this program.

I develop this program in ambience of LabVIEW, it will work for expansion options of using usual data acquisition boards, because usual data acquisition boards don't have HW phase loop lock (PLL) on it.

Phase loop lock is back loop control circuit which generates output signal with the same frequency and phase (eventually multiple of frequency) as the signal input. This work uses the principle of PLL to generate output signal witch corresponding to the multiple frequency of input signal.

I use principle of PLL for optimization of sampling unknown periodic input signal – abiding integer number of period of measured signal in measured data for next processing this digitized signal for example application of FFT.

Key Words

Frequency, phase, Tw (time window), phase loop lock (PLL), fast Fourier transform (FFT), the measuring board, data acquisition, sampling frequency, LabVIEW

Obsah

Seznam tabulek

Seznam grafů

Seznam obrázků

1 Úvod

Snahou každého člověka je zjednodušení a zvýšení efektivity prováděné činnosti. Lidé se vždy snažili využít dostupných možností pro zjednodušení a mechanizaci prováděných prací a tím snižování nároků na člověka. K tomuto účelu využívali mechanické stroje od jednoduchých až po složité mechanizmy.

V dnešní době sice využíváme stále stejné základní fyzikální principy, ale k pohonu a řízení využíváme elektrické systémy. Pro řízení procesů a udržení kvality výroby je nezbytné získávat informace jak o vstupních parametrech procesu, tak o stavu výstupu. Touto problematikou se zabývá měření.

Při měření rozlišujeme 2 druhy signálů – spojitý a diskrétní. V dnešní době počítačové techniky provádíme veškeré zpracování signálů pouze v podobě diskrétních elektrických signálů, tedy pro měření spojitého signálu je nutné provézt diskretizaci A/D převodníkem (převádí analogovou hodnotu na digitální), to sebou ale nese mnoho úskalí.

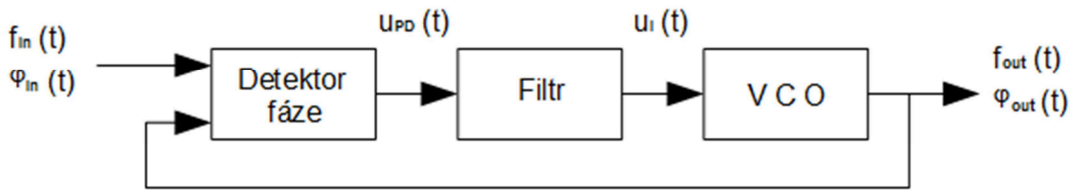
Pro měření konstantních nebo stále se opakujících signálů není problém předvídat jejich vlastnosti, a tedy nastavit parametry A/D převodníku tak, že dostaneme informaci o měřeném signálu v nejlepší možné kvalitě. Problém nastává u signálů s proměnnými vlastnostmi, v tomto případě je nutné reagovat na změny vlastností signálu a měnit nastavení A/D převodníku pro udržení kvality naměřených dat.

Touto problematikou se zabývá tato bakalářská práce. Jejím cílem je optimalizovat měření spojitého analogového signálu proměnné frekvence pro následné zpracování například FFT analýzou. V kapitolách práce se věnuji vývoji programu fázového závěsu a popisem vlastností jednotlivých částí.

2 Popis PLL

PLL (phase locked loop) je regulační obvod složený z oscilátoru a fázového detektoru, porovnávající fázový posun vstupního signálu f_{in} se signálem výstupu oscilátoru f_{out} pomocí zpětné vazby. Žádanou veličinu zde představuje frekvence a fáze vstupního signálu f_{in}, φ_{in}, regulovanou veličinu frekvence a fáze výstupního signálu f_{out}, φ_{out}. Žádaný signál je v detektoru fáze porovnáván s regulovaným a jejich rozdíl u_{PD}, představující akční veličinu, přiveden na vstup filtru typu dolní propust kvůli odstranění šumu a stabilitě regulačního obvodu. Výstupní signál filtru u_l (ladící napětí), prezentující stále akční veličinu, vstupuje do oscilátoru VCO (napětím řízeného oscilátoru) představujícího regulovanou soustavu, kde změnou frekvence oscilátoru (akční zásah) docílí synchronizaci fází φ_{in} a φ_{out} (regulační odchylka = 0). Blokové schéma popsaného principu ukazuje obrázek č. 1.

Tento princip je obecně využíván k synchronizaci, demodulaci a frekvenční syntéze signálů v mnoha oblastech elektroniky- například v rádiových přijímačích frekvenčně modulovaného signálu. Zde připojením vstupu zpětné vazby fázového detektoru k pevně nastavenému oscilátoru o frekvenci přijímané stanice a vstupu žádané veličiny k pásmové propusti (signál z antény), sloužící k filtrování okolních rozhlasových stanic, dostaneme na výstupu filtru u_l žádaný nf audio signál (zjednodušený příjem monofonního frekvenčně modulovaného signálu).



Obr. 1: Blokové schéma fázového závěsu

V této bakalářské práci využívám principy fázového závěsu k optimalizaci vzorkování neznámého periodického signálu. Z měřeného signálu počítám frekvenci a na základě této hodnoty měním vzorkovací frekvenci tak, abych ve vyčítaných naměřených datech obdržel vždy celočíselný násobek periody měřeného signálu.

2.1 Popis měřících karet

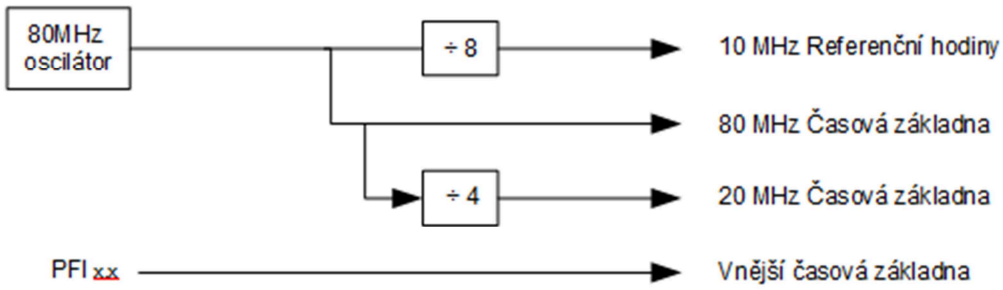
HW pro tuto bakalářskou práci představují měřící karty sloužící ke sběru případně generování analogových a digitálních dat pro jejich následné zpracování. Měřící karta je doplňkovým zařízením počítače s rozhraním závislým na typu karty. Karty s rozhraním PCI představují doplňkový HW stolního PC se sběrnicí PCI. Karty s rozhraním USB jsou propojitelné jak se stolním PC, tak s notebookem či jiným HW s rozhraním USB. Protože zpracování signálů probíhá v současné době výhradně digitální cestou, obsahují měřící karty A/D převodníky pro měření, případně D/A převodníky pro generování analogového signálu, digitální vstupy a digitální výstupy.

PCI:

K měření v laboratoři používáme měřící kartu PCI 6221 firmy National Instruments. Jde o rozšiřující kartu stolního PC s rozhraním PCI. Karta disponuje dále uvedenými periferiemi:

- 16 x AI Rozlišení A/D převodníku 16 bitů Vzorkovací frekvence 250kS/s
- 2 x AO Rozlišení D/A převodníku 16 bitů Vzorkovací frekvence 833kS/s
- 24 x DI / DO
- 2 x counter / timer 32 bitů

Časování karty PCI je řízeno buď z vnitřního stabilního 80MHz oscilátoru a frekvencemi odvozenými jeho dělením nebo hodinovým signálem z vnějšího vstupu nebo softwarově časovacími signály operačního systému. Možnosti časování ukazuje obrázek č. 2.



Obr. 2: Zdroje hardwarového časovacího signálu

USB:

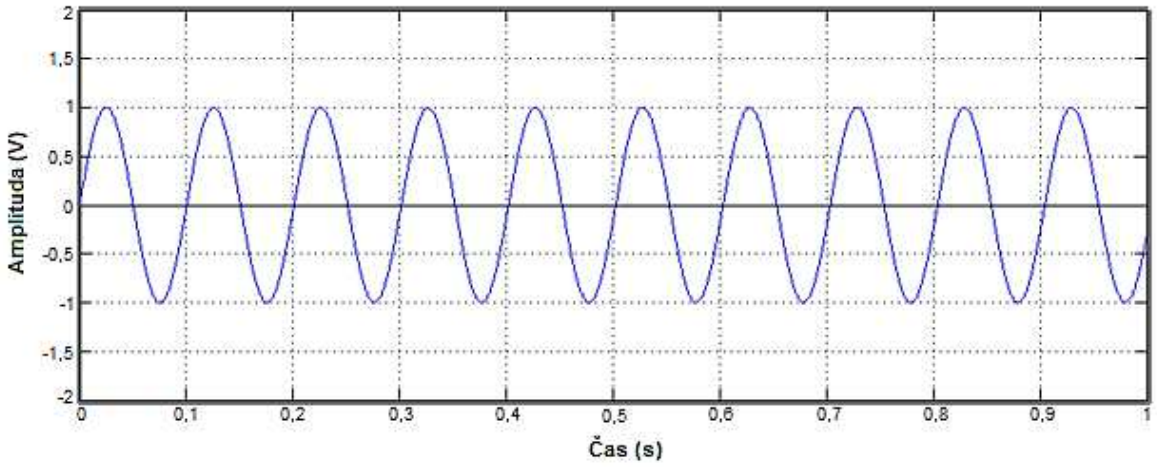
Další měřící kartou, pro kterou bude určen vyvíjený software, je USB6210 taktéž firmy National Instruments. Jedná se o externí měřící kartu komunikující přes rozhraní USB. Tato karta má o něco horší výbavu než předchozí a to:

- 16 x AI Rozlišení A/D převodníku 16 bitů Vzorkovací frekvence 250kS/s
- 4 x DI
- 4 x DO
- 2 x counter / timer 32 bitů

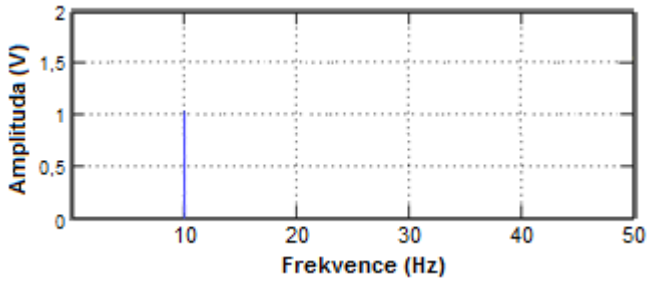
Časování karty USB je řízeno obdobně jako v předchozím případě buď z vnitřního 80MHz oscilátoru případně z odvozeného 20MHz signálu nebo hodinovým signálem z vnějšího vstupu.

2.2 Popis FFT

FFT (Fast Fourier Transform) je efektivní algoritmus pro převod signálu z časového spektra do frekvenčního (rozkládá komplexní signál na součet jednotlivých sinových a kosinových složek). Zobrazení jednotlivých složek ve frekvenční oblasti nazýváme amplitudovým frekvenčním spektrem, zobrazení složek signálu a jejich počátečních fází nazýváme fázovým frekvenčním spektrem signálu. Dle rozsahu analyzovaného spektra frekvencí dělíme FFT na jednostranné spektrum o frekvenčním rozsahu 0 až ∞Hz, v tomto případě amplitudy spektrálních čar amplitudového spektra odpovídají amplitudám sinových a kosinových složek, a dvoustranné spektrum o frekvenčním rozsahu -∞ až +∞, v tomto případě jsou amplitudy spektrálních čar ½ amplitudy sinových a kosinových složek.
Z teorie tedy vyplývá, že provedením FFT na čistě sinusovém signálu (například o frekvenci f = 10Hz a amplitudě 1V graf č. 1) získám při jednostranném měření spektra v amplitudovém spektru 1 spektrální čáru o amplitudě rovné amplitudě sinusového signálu na frekvenci odpovídající frekvenci tohoto signálu (graf č. 2).



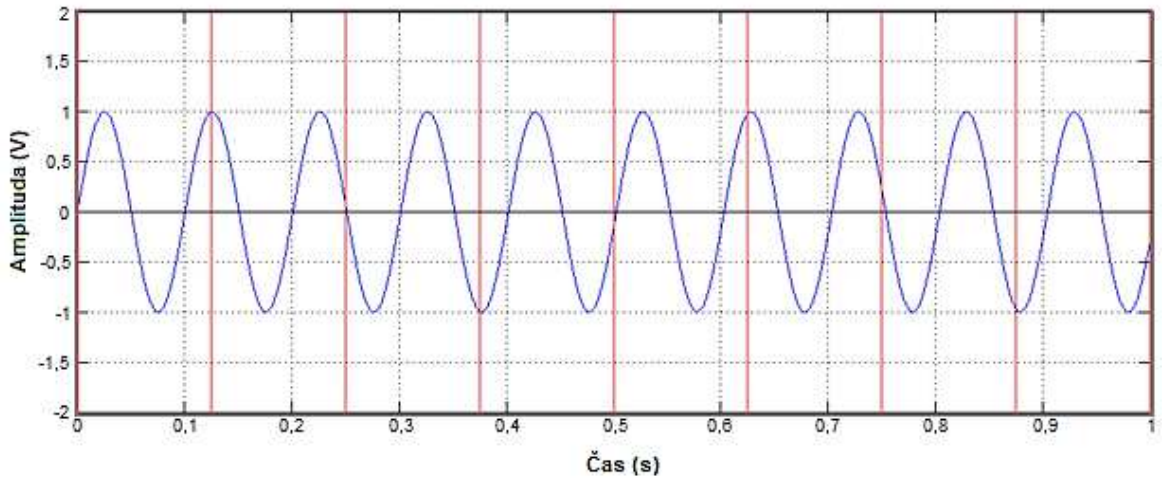
Graf 1. U = f(t) Příklad sinusového signálu



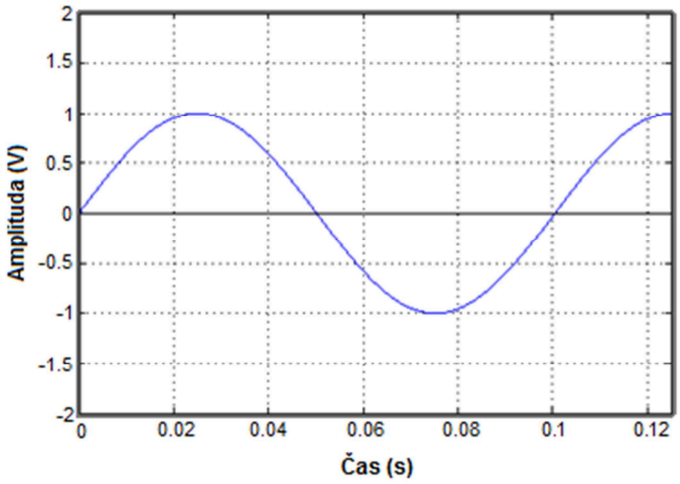
Graf 2. U = f(f) Amplitudové spektrum ideálního sinusového signálu

Výše zmíněné je pravda ovšem pouze za předpokladu provádění FFT analýzy z nekonečně dlouhého sinusového signálu (analýza spojitého signálu v časové oblasti). Protože v reálu není možno analyzovat nekonečně dlouhý signál, musíme tento signál rozdělit na časová okna (Time Window zkráceně Tw), která již lze postupně zpracovat. Tím se dostáváme k problému celistvosti period v daném Tw. Protože zpracování měřeného signálu probíhá digitální cestou, je třeba spojitý analogový signál převézt na diskrétní (provézt digitalizaci). Tento proces obstarává A/D převodník měřící karty dle nastavených parametrů. Těmito parametry jsou vzorkovací frekvence (rychlost získávání jednotlivých vzorků ze spojitého signálu) a počet vzorků (skupina vzorků o daném počtu (Tw) zaslaná ke zpracování počítači). U stabilních signálů o stálé frekvenci je možno nastavit parametry měření tak, aby Tw obsahovalo přesně daný počet period měřeného signálu a lze tedy provézt FFT analýzu sice signálu omezené délky, ale celočíselného násobku periody přesně. Vzorkujeme tedy konstantní rychlostí a vyčítáme konstantní počet vzorků nastavený dle parametrů měřeného signálu. Takovémuto způsobu vzorkování říkáme ekvidistantní vzorkování (vzorkovací frekvence je v celém změřeném časovém okně stejná - časová prodleva mezi vzorky je konstantní). Problém nastává u signálů neznámé nebo nestabilní frekvence, kde musíme analyzovat frekvenci měřeného signálu a následně nastavit optimální vzorkovací frekvenci (u stabilních signálů viz graf č. 3,

kde modrá křivka zobrazuje měřený signál, červená hranice jednotlivých Tw) nebo neustále měřit frekvenci vstupního signálu a zpětnou vazbou plynule měnit vzorkovací frekvenci (u nestabilních signálů). Zde vzniká nebezpečí změny rychlosti čtení v průběhu vyčítání vzorků. Takto naměřené Tw může obsahovat více vzorkovacích frekvencí, tzn. dochází k neekvidistantnímu vzorkování. Z takovýchto dat nejsem schopen analyzovat správnou frekvenci měřeného signálu.

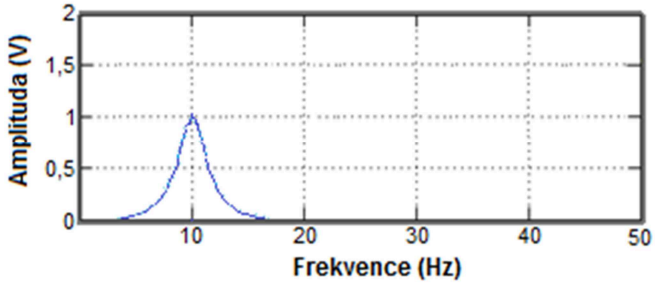


Graf 3. U = f(t) Nekorektně navzorkovaný signál (červené čáry zobrazují hranici jednotlivých Tw)



Graf 4. U = f(t) Tw při navzorkování necelistvého počtu period signálu

Při provedení FFT analýzy Tw podobného grafu č. 4 již neobdržíme 1 spektrální čáru, nýbrž celé spektrum frekvenčních čar v okolí frekvence původního signálu (dojde k tzv. prosakování spektra). Tedy takto obdržené výsledky FFT analýzy neodpovídají měřenému signálu. Viz graf č. 5.



Graf 5. U = f(f) FFT nekorektně navzorkovaného signálu

3 SW realizace PLL

Základní myšlenkou vyvíjeného programu je optimalizovat proces sběru dat pro výpočet korektního FFT pro další zpracování.
Je třeba získat celistvý počet period měřeného signálu k dalšímu zpracování. Tohoto lze dosáhnout dvěma způsoby:

1. Měnit frekvenci vzorkování při konstantním počtu měřených vzorků
2. Měnit počet vzorků při konstantní vzorkovací frekvenci

U obou způsobů řešení je problém v tom, že pro správné nastavení rychlosti vzorkování potřebujeme znát aktuální frekvenci měřeného signálu a tedy Tw, ze kterého tuto frekvenci počítáme je potenciálně navzorkováno nekorektně (je možné, že bylo vzorkováno s nesprávnou vzorkovací frekvencí).

U prvního způsobu je nevýhoda, že analýza frekvence probíhá z celého Tw a je tedy nutné dodržet konstantní rychlost vzorkování v celé skupině vzorků. V případě nedodržení této podmínky analyzovaná frekvence neodpovídá skutečné a dojde k rozkmitání regulátoru PLL. Na druhou stranu změna vzorkovací frekvence má malý vliv na výsledné Tw viz rovnice č. 2.

U druhého způsobu řešení opět platí, že aktuálně analyzované Tw je navzorkováno nekorektně a pro udržení celistvého počtu period je nutné nalézt počátek a konec periody měřeného signálu a necelistvé okraje odstranit. Tím dojde ke ztrátě části naměřených dat, což může mít za následek ztrátu informací v měřeném signálu. Dalším omezením je maximální měřitelná frekvence, ta musí splňovat Nyquistův teorém, tj. maximální hodnota odpovídá polovině vzorkovací frekvence. Další nevýhodou je mnohem větší vliv změny počtu vzorků na výsledné Tw než v případě řízení vzorkovací frekvence viz rovnice č. 3.

Názornou ukázkou vlivu jednotlivých způsobů řízení jsou následující rovnice, kde f_{vz} představuje vzorkovací frekvenci, f_{sig} představuje frekvenci měřeného signálu, n počet vzorků na 1 periodu měřeného signálu. Výpočet počtu vzorků na periodu měřeného signálu ukazuje rovnice č. 1. Zvýšením vzorkovací frekvence o 1 dojde ke zvětšení počtu vzorků na periodu o $1 / f_{sig}$, dle rovnice č. 2. Zvětšením počtu vzorků o 1 musím zvýšit vzorkovací frekvenci o frekvenci měřeného signálu viz rovnice č. 3.

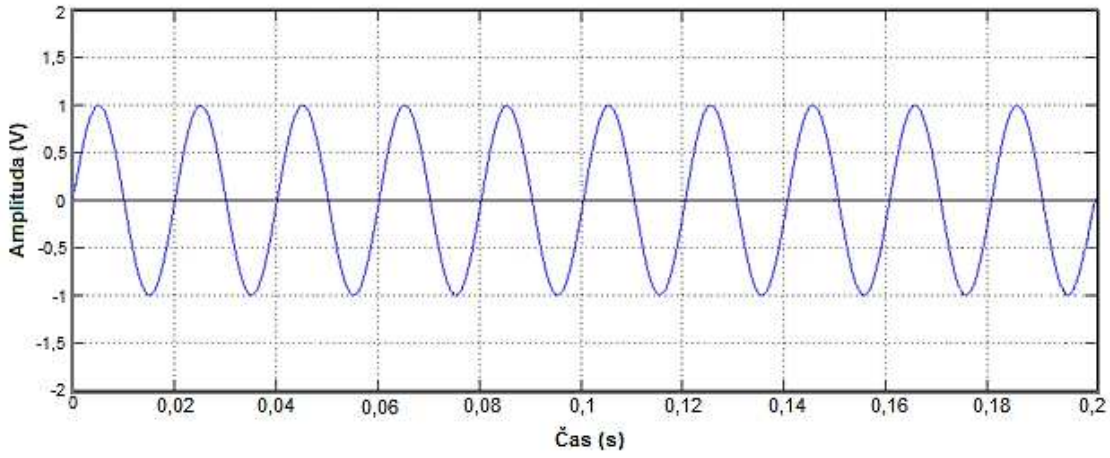
$$\frac{f_{vz}}{f_{sig}} = n \tag{1.}$$

$$\frac{\mathrm{f_{vz}} + 1}{\mathrm{f_{sig}}} = \mathrm{n} + \frac{1}{\mathrm{f_{sig}}} \tag{2.}$$

$$\mathrm{n} + 1 = \frac{\mathrm{f_{vz}} + \mathrm{f_{sig}}}{\mathrm{f_{sig}}} \tag{3.}$$

3.1 Praktická realizace a vývoj programu

Program je určen pro měření signálů napájecí sítě tj. 50Hz sinusového signálu. Pro optimální zpracování naměřených dat vyčítám 10 period tohoto signálu rychlostí odpovídající 20 násobku frekvence signálu, tj. 1kHz. Na základě těchto požadavků vyčítám data po 200 vzorcích (velikost Tw = 200). Příklad takto změřeného Tw ukazuje graf č. 6. Protože předpokládám měření nízkofrekvenčního signálu (50Hz) kartou sběru dat, je vhodnější jít směrem prvního způsobu řízení tj. pro udržení celočíselného násobku period měřeného signálu měnit vzorkovací frekvenci při konstantním počtu vyčítaných vzorků. Výhodou je jednak zachování všech naměřených dat, dále menší vliv změny rychlosti vzorkování na výsledné Tw a v případě požadavku na měření vyšších frekvencí omezení pouze HW měřící karty.



Graf 6. Tw optimálně změřeného 50Hz signálu

Vliv obou způsobů řízení na 50Hz signál je následující: změna vzorkovací frekvence o 1Hz odpovídá

$$\frac{f_{vz}+1}{f_{sig}} = n + \frac{1}{f_{sig}} \quad \gg \quad \frac{f_{vz}+1}{50} = n + \frac{1}{50} = n + 0{,}02 \tag{4.}$$

tj. změně počtu vzorků Tw o 0,02 vzorku. Naopak změně počtu vzorků o 1 vzorek odpovídá

$$n + 1 = \frac{f_{vz}+f_{sig}}{f_{sig}} \quad \gg \quad n + 1 = \frac{f_{vz}+50}{50} \tag{5.}$$

tj. změna vzorkovací frekvence o 50Hz.

3.1.1 Možnosti změny rychlosti čtení

Karty sběru dat firmy National Instruments umožňují nastavení hodinového signálu více způsoby. Nejpoužívanější způsob je nastavení vlastnosti uzlu DAQmx (Property DAQmx Timing Rate). Tomuto uzlu zadám žádanou rychlost hodinového signálu a on sám následně provede výběr nejvhodnější časové základny a nastavení vhodného dělícího poměru vnitřní děličky. Další možností je odvodit časovací signály z jiného zdroje hodinového signálu (z jiné vzorkovací frekvence – Sample clock) a to buď z analogových linek AI Sample clock, AO Sample clock nebo z digitálních linek DI Sample clock, DO Sample clock. Další možností nastavení časovacího signálu je odvodit jej z vnějšího signálu přivedeného k digitální lince PFIx.

Měřící karta PCI6221 umožňuje řídit rychlost čtení AI následujícími způsoby (detailně popsáno dále):
Property DAQmx Timing Rate
AO Sample clock
DI Sample clock
DO Sample clock
Externí vstup (PFIx)

Měřící karta USB6210 nedisponuje analogovým výstupem, má tedy omezenější možnosti:
Property DAQmx Timing Rate
Externí vstup (PFIx)

3.1.2 Využití uzlu „Property DAQmx Timing"

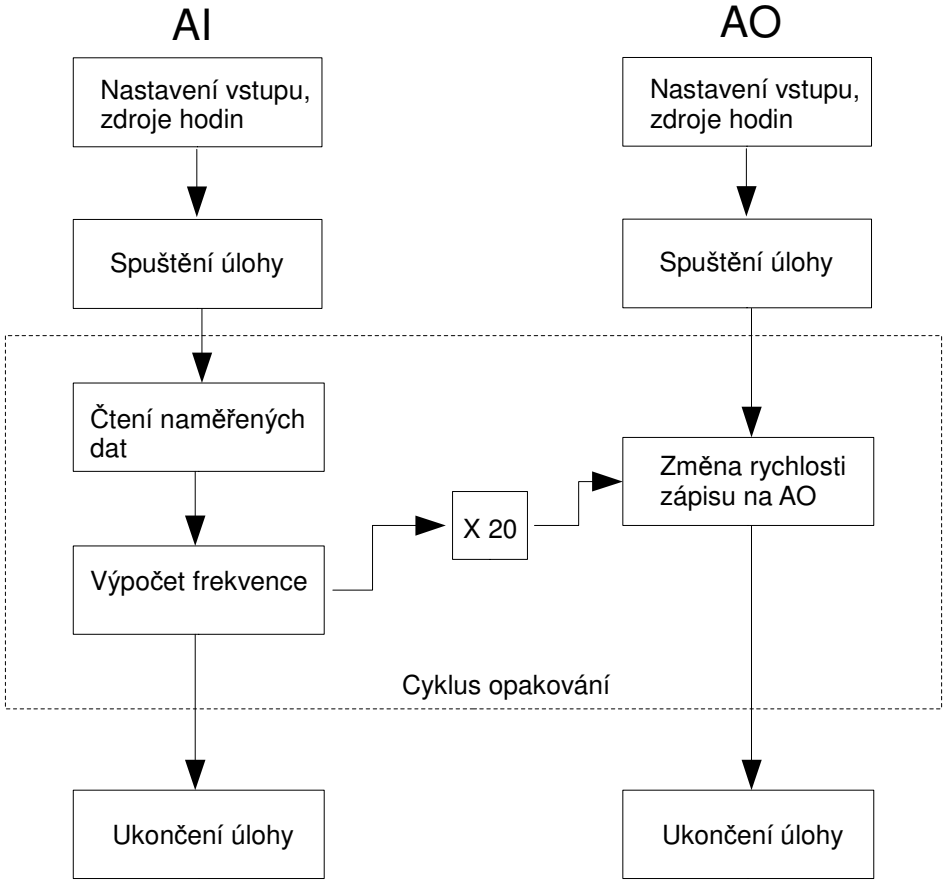
V LabVIEW lze měnit rychlost čtení analogového vstupu (AI - analog input) v uzlu vlastností DAQmx Timing změnou rychlosti čtení (Rate). Tato možnost platí ovšem pouze při změně před spuštěním samotného procesu čtení, za běhu tuto rychlost v Timing Property Node již nelze měnit a zadáním nové hodnoty rychlosti čtení dojde k chybě (vlastnost nemůže být nastavena, dokud

úloha běží). Jedinou možností opětovné změny rychlosti vzorkování je restartovat úlohu čtení AI s novou hodnotou.

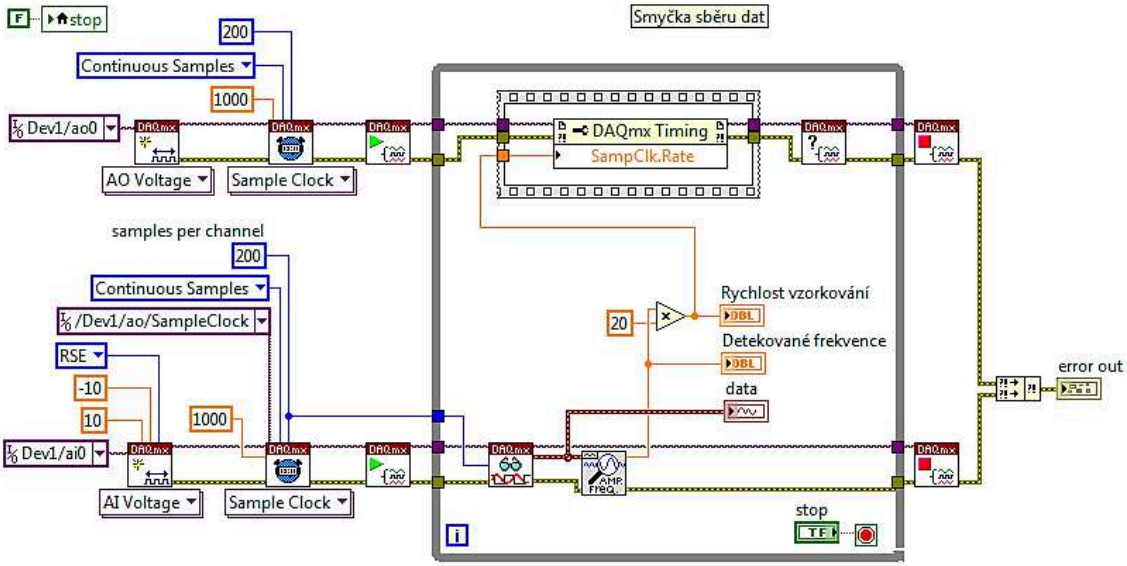
Takovýto způsob není vhodný z důvodu časových prodlev mezi ukončením a opětovným spuštěním úlohy čtení a tím pádem ztrátou velkého množství dat. Měření tedy není kontinuální.

3.1.3 Využití analogového výstupu jako zdroje hodinového signálu

Další možností je přepnout vzorkovací frekvenci na jiný zdroj hodinového signálu než je generátor na měřící kartě (zdroj volíme pomocí Source v uzlu DAQmx Timing). Zde mi pomohly příklady z diskuzních fór uživatelů LabVIEW a já použil jako zdroj hodinového signálu vzorkovací signál úlohy analogového výstupu (AO Sample Clock). AO má podobně jako digitální linky zajímavou vlastnost a to, že lze za běhu procesu zapisování hodnot na analogový výstup měnit rychlost obnovy těchto hodnot. Takto mohu v úloze analogového vstupu v uzlu DAQmx Timing použít vzorkovací signál analogového výstupu jako zdroj hodin, kdy při změně frekvence obnovy AO v uzlu vlastností DAQmx Timing (Rate v DAQmx Timing Property Node) dojde zároveň ke změně rychlosti čtení AI. Takto vznikla první nejjednodušší verze programu sw PLL, blokové schéma ukazuje obrázek č. 3, ukázku kódu znázorňuje obrázek č. 4. Na CD se tento kód nachází v adresáři Programy, zdrojový kód v LabVIEW je pojmenován **PLL using AO.vi** .



Obr. 3: Blokové schéma využití AO jako zdroje hodinového signálu

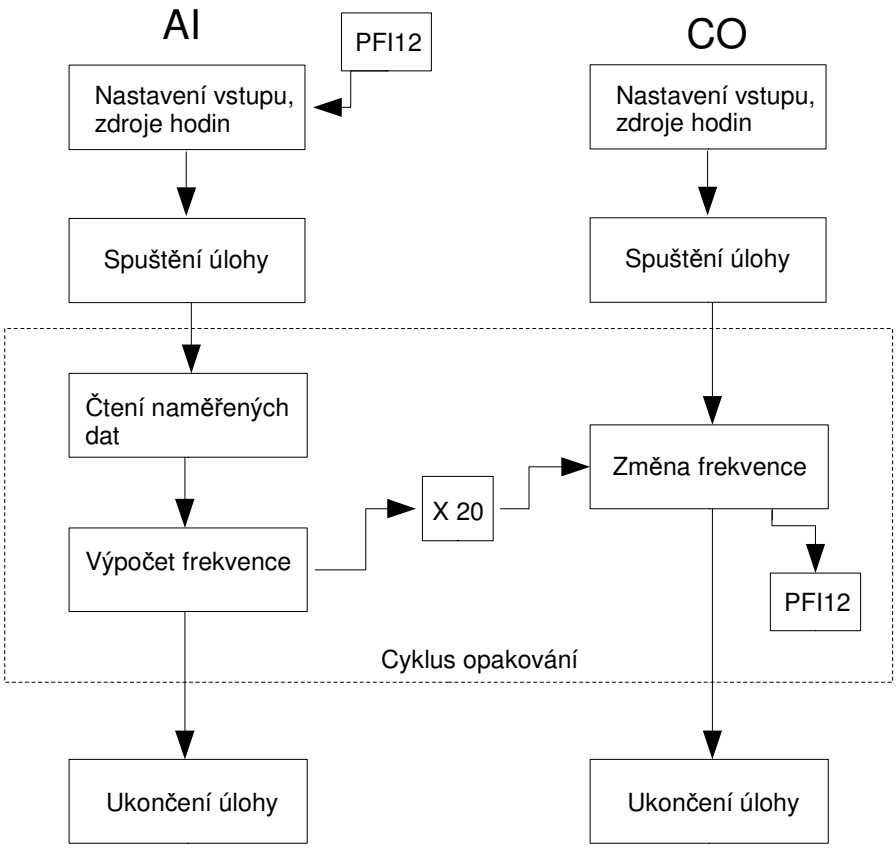


Obr. 4: Zdroj hodin = vzorkovací frekvence AO

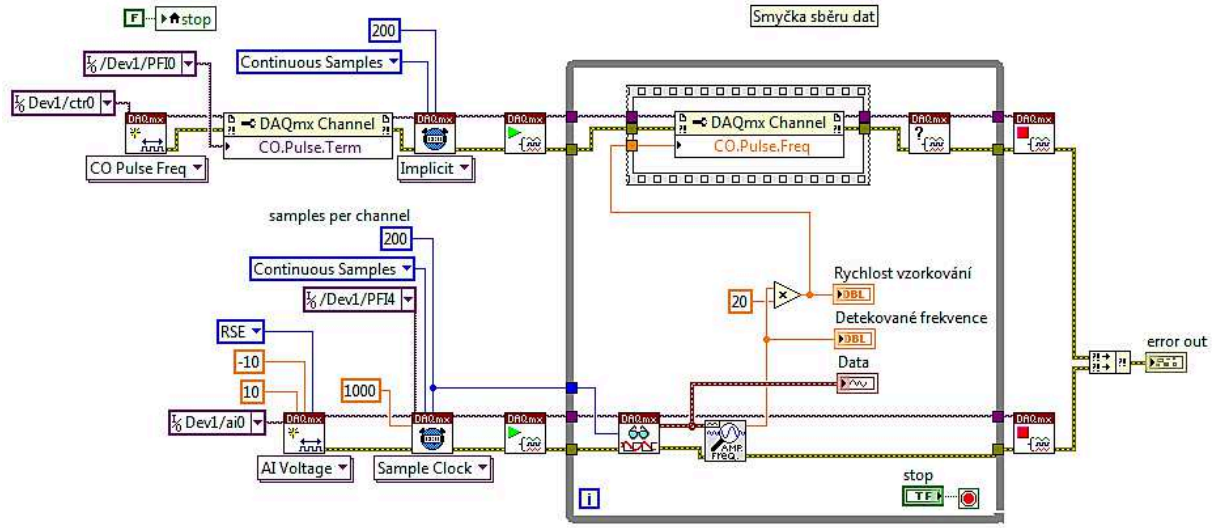
Tato verze programu je již funkčním programem PLL, nicméně projevuje se množstvím nedokonalostí jako je chyba při odpojení měřeného signálu, kmitání vzorkovací frekvence a rozkmitání regulace PLL vlivem zpoždění regulační smyčky.

3.1.4 Využití externího vstupu jako zdroje hodinového signálu

Protože měřící karta USB 6210 nedisponuje analogovým výstupem, a tedy předchozí navrhované řešení je pro ni nevhodné, musel jsem vyzkoušet další způsob řízení PLL.
Pří této konfiguraci musím vzorkovací signál buď generovat na dalším HW a fyzicky přivést k měřící kartě nebo využít další funkce karty a to generování signálu proměnné frekvence. Použil jsem časovač ctr0 jako generátor signálu proměnné frekvence „Counter Output Pulse Frequency". Takto generovaný signál posílám na výstup měřící karty PFI12, ze kterého jej v úloze řízení čtení analogového vstupu (AI) zpětně čtu a využívám jako časovou základnu pro vzorkování AI. Nyní tedy měním frekvenci generovaného signálu v úloze Counter output v uzlu vlastností DAQMX Channel Property Node pomocí vstupu Counter Output Pulse Frequency a tento signál posílám na výstup PFI12. V úloze AI je PFI12 nastaven jako zdroj hodinového signálu (Source v uzlu DAQmx Timing) a využívám jej pro řízení rychlosti čtení AI, blokové schéma ukazuje obrázek č. 5. Takto vznikla druhá verze programu použitelná pro obě měřící karty (obrázek č. 6). Na CD se uvedený kód nachází v adresáři Programy, zdrojový kód v LabVIEW je pojmenován **PLL using CO.vi**.



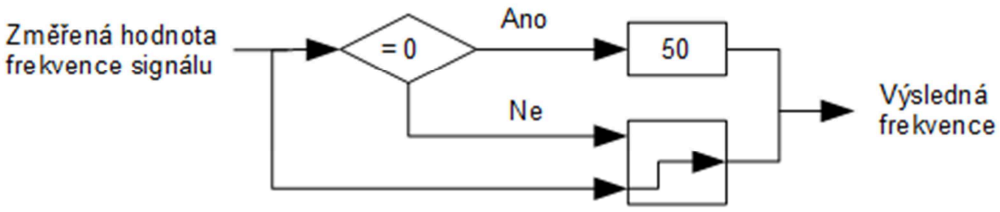
Obr. 5: Blokové schéma využití CO jako zdroje hodinového signálu



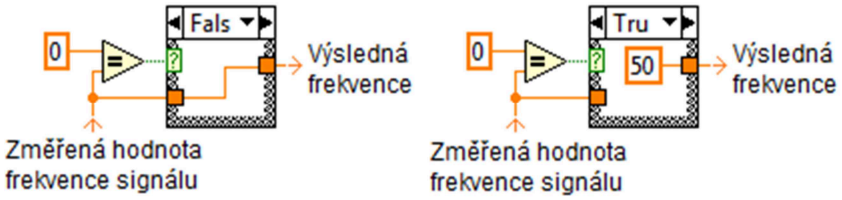
Obr. 6: Zdroj hodin - externí vstup PFI12

3.1.5 Odstranění problému s odpojeným vstupním signálem

Odpojením měřeného signálu nebo uzemněním měřícího vstupu karta změří konstantní signál 0V, který program počítající frekvenci (sub VI Extract Single Tone Information) vyhodnotí jako 0Hz, což blok, provádějící změnu generované frekvence, ohodnotí chybou jako hodnotu mimo povolený rozsah (frekvence generovaného signálu = 0Hz). Možnou nápravou je čekat na hodnotu měřeného signálu 0Hz a nahradit ji jinou hodnotou dříve, než ji blok provádějící změnu frekvence generovaného signálu zpracuje, viz obrázek č. 7.



Obr. 7: Blokové schéma opravy chyby s odpojeným vstupem



Obr. 8: Oprava odpojeného vstupu a) korektní frekvence, b) vstup odpojen

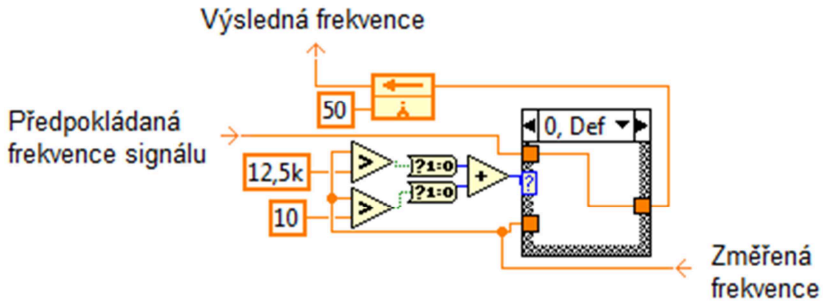
Tímto řešením po odpojení měřeného signálu kartě tvrdím, že měří 50Hz předpokládaný signál. Ukázku kódu znázorňuje obrázek č. 8. Program tedy nejenže nevyhodnotí chybu a neukončí se, ale zároveň vzorkuje analogový vstup vzorkovací frekvencí 1kHz, a tedy po připojení signálu do Nyquistovy frekvence (do 500Hz) nedojde k aliasingu, ale v dalším průběhu smyčky PLL ke korektní analýze frekvence přivedeného signálu.

3.1.6 Zajištění korektního chování po překročení limitů vzorkovací frekvence karty

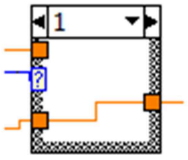
V souvislosti s předchozím problémem jsem se při testování dostal k problému při měření vyšších frekvencí a to překročení maximální povolené vzorkovací frekvence, což podobně jako při odpojení měřeného signálu skončí chybou - hodnota mimo povolený rozsah. Provedl jsem tedy doplnění předchozí podmínky. Pokud frekvence měřeného signálu f_{sig} překročí 12,5kHz, což odpovídá frekvenci vzorkování f_{VZ} 250kHz, nastavím hodnotu signálu f_{sig} zpět na 12,5kHz (obrázek č. 11). Výpočet vzorkovací frekvence f_{VZ} popisuje rovnice č. 6 jako $a = 20$ násobek frekvence měřeného signálu f_{sig}.

$$a \cdot f_{sig} = f_{VZ} \qquad 20 \cdot 12{,}5 \cdot 10^3 = 250 \cdot 10^3 Hz \qquad (6.)$$

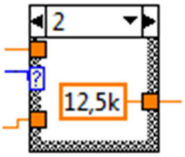
Podobný problém nastává v oblasti nízkých frekvencí po připojení signálu k měřící kartě za běhu programu. Při odpojeném vstupním signálu karta díky kontinuálnímu vzorkování rychlostí 1kHz měří šumové signály o různých frekvencích a dochází k problému. Pokud je frekvence šumu nízká, měřící karta provádí vzorkování nízkou frekvencí a po přivedení žádaného signálu se tento signál zrcadlí do nižších frekvencí (dojde k jevu, známém jako aliasing, jde o porušení Nyquistova vzorkovacího teorému, kdy frekvence měřeného signálu je vyšší než polovina vzorkovací frekvence). V tomto případě program není schopen tento stav rozlišit a provádí vyhodnocení nekorektního signálu. Tento problém jsem odstranil nastavením minimální měřené frekvence na hodnotu 10Hz, kdy pokles pod tuto hodnotu vlivem šumu či aliasing signálu způsobí nahrazení informace o měřené frekvenci hodnotou 50Hz, případně jinou předpokládanou hodnotou frekvence signálu, a nezpůsobí snížení rychlosti vzorkování pod 200Hz, nýbrž nastaví vzorkování na optimální vzorkovací frekvenci předpokládaného signálu 1kHz a lze tedy bez obav připojit měřený signál do frekvence 500Hz. Praktická realizace této části kódu je znázorněna v sérii obrázků a to č. 9, č. 10 a č. 11, tento program se nachází v příloze na CD v adresáři Programy, pod názvem **PLL using CO zajištění korektního chování při překročení limitů nebo odpojení vstupního signálu.vi** .



Obr. 9 Zajištění korektní f_{vz} při nedodržení rozsahu měřených frekvencí (měřená frekvence < 10Hz)



Obr. 10 Měřená frekvence je v rozsahu



Obr. 11 Měřená frekvence mimo rozsah (fvz > 12,5kHz)

Přepracoval jsem předchozí část programu Odstranění problémů s odpojeným vstupním signálem a doplnil podmínky tak, že při analýze signálu o frekvenci menší jak 10Hz předpokládám měření šumu případně aliasing signálu a vnucuji programu hodnotu 50Hz (obrázek č. 9), při překročení maximální měřené frekvence 12,5kHz nahrazuji informaci o frekvenci měřeného signálu hodnotou 12,5kHz, udržuji tak vzorkování na maximální hodnotě tj. 250kHz (obrázek č. 11).

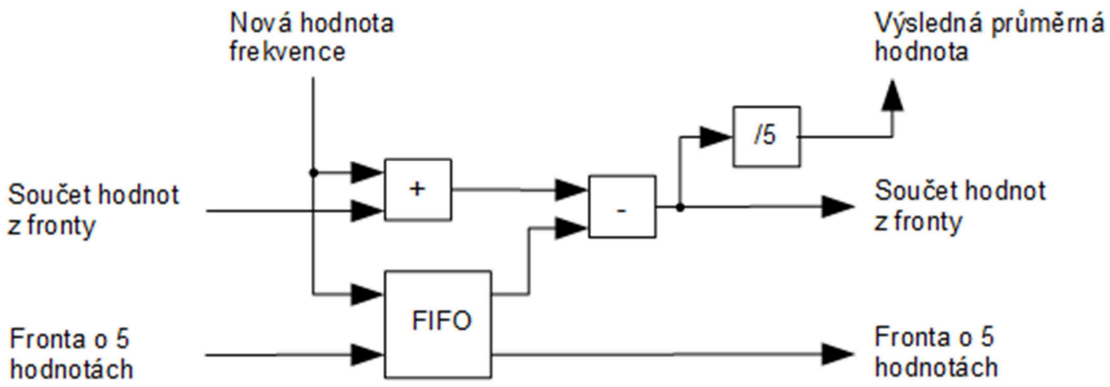
3.2 Filtrace kmitání analyzované frekvence měřeného signálu

V této fázi je již program použitelný pro obě měřící karty, má ochranu proti překročení rozsahu měřených frekvencí a nastal čas řešit problémy s nestabilitou frekvence měřeného signálu, která se projevila při testování. Ověřil jsem základní funkčnost a zjistil, že analyzovaná frekvence přivedeného signálu kmitá okolo předpokládané hodnoty frekvence tohoto signálu. To může být způsobeno několika faktory - 1. nestabilita frekvence měřeného signálu + rušení, 2. vyčtena méně jak 1 perioda měřeného signálu. Jako možná náprava tohoto problému mne napadlo filtrovat získané hodnoty a odstranit tak vliv naměřených odchylek od skutečné hodnoty. Postupným vývojem jsem došel na dvě možnosti odstranění tohoto problému a to filtrace změřené frekvence:

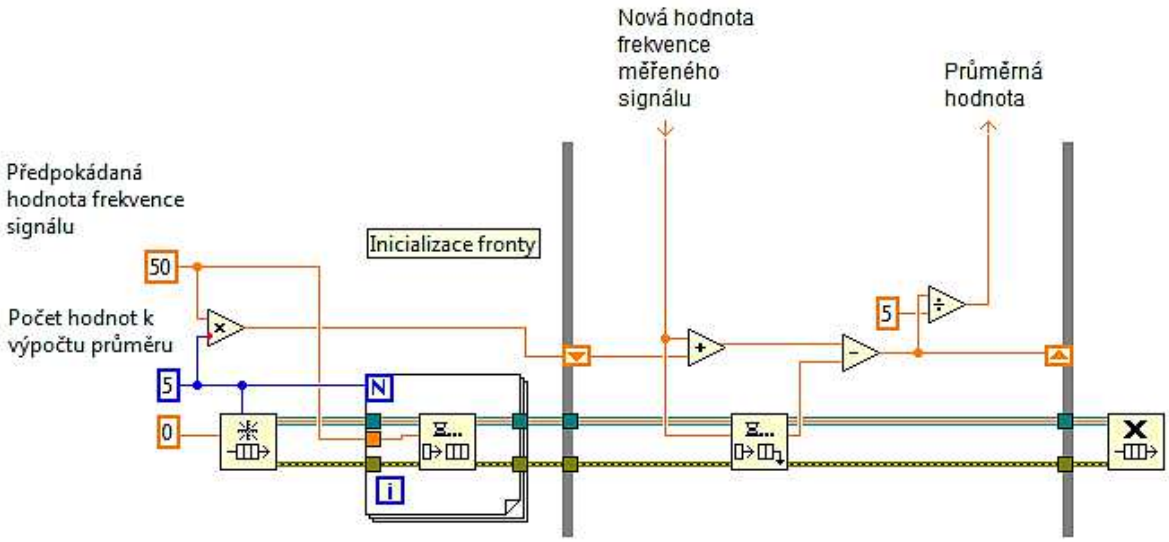
1. Výpočet průměrné hodnoty z předchozích změřených hodnot
2. Zpomalení změny frekvence signálu při malých odchylkách oproti předchozí

3.2.1 Výpočet průměrné hodnoty z předchozích změřených hodnot

Základem tohoto způsobu filtrace je fronta (Queue) uchovávající konečný počet předchozích hodnot frekvence měřeného signálu pro výpočet průměrné hodnoty. Abych při každém cyklu nevyčítal hodnoty z fronty a nepočítal průměr znovu (a tím nezpomaloval cyklus výpočtu), uchovávám součet hodnot fronty odděleně v paměti (shift registr), takže při opětovném provádění cyklu výpočtu průměrné hodnoty, přičtu k celkovému součtu novou hodnotu, odečtu poslední hodnotu z fronty (po překročení nastaveného počtu vzorků fronta automaticky po přijetí nové hodnoty uvolní nejstarší) a podělím konstantou odpovídající počtu vzorků ve frontě. Blokové schéma znázorňuje obrázek č. 12, ukázku kódu znázorňuje obrázek č. 13. Uvedený kód se nachází v příloze na CD, program v LabVIEW je uložen pod názvem **PLL using CO filtrace kmitání měřené frekvence.vi** .



Obr. 12: Blokové schéma výpočtu průměrné hodnoty



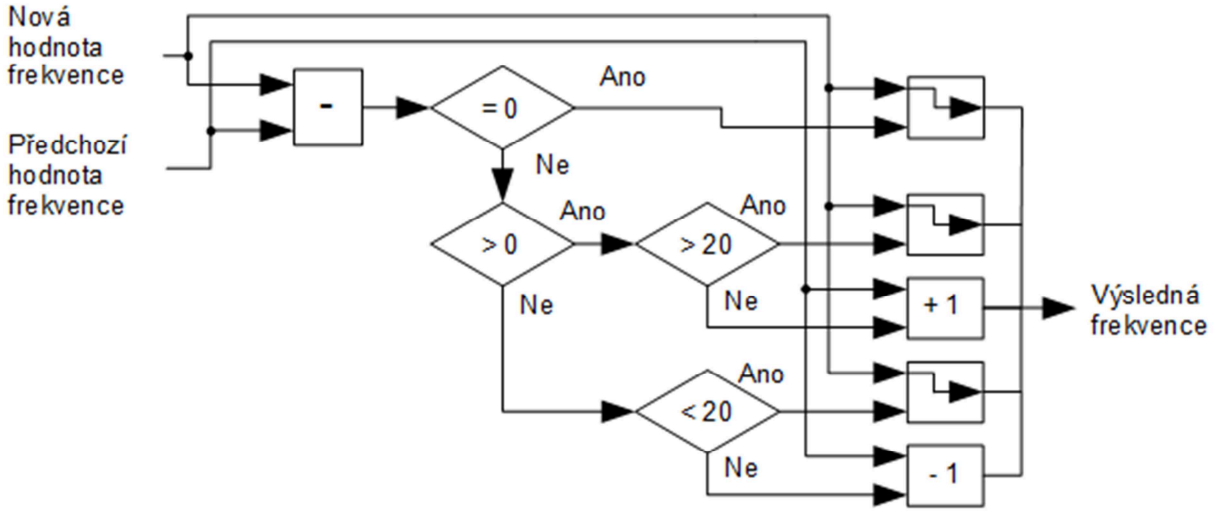
Obr. 13: Realizace výpočtu průměrné hodnoty

3.2.2 Zpomalení změny frekvence signálu při malých odchylkách oproti předchozí

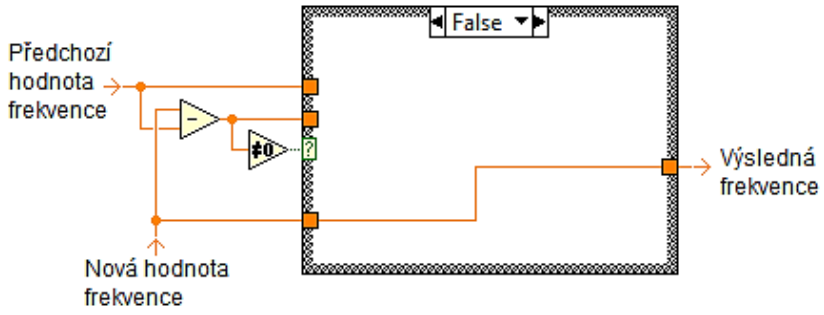
Při tomto způsobu filtrace porovnávám novou hodnotu frekvence měřeného signálu s předchozí a na základě velikosti rozdílu a vhodnou kombinací podmínek zajišťuji, že změna frekvence vzorkování neprobíhá skokově, ale po malých krocích.

Pokud je rozdíl změřené frekvence signálu z aktuálního Tw a předchozí hodnoty 0, pak výsledná hodnota frekvence signálu a po přepočtu frekvence vzorkování zůstává stejná. Pokud je rozdíl frekvencí signálů menší jak 20Hz, pak k předchozí hodnotě přičtu 1 a každým průběhem cyklu snižuji rozdíl až na 0. Pokud je rozdíl větší jak 20Hz, pak předpokládám, že ke změně došlo záměrně, a pro urychlení doby ustálení pošlu tuto vyšší hodnotu frekvence signálu dál, tzn. výsledná hodnota frekvence signálu = aktuálně změřená. Stejný děj probíhá v případě záporného rozdílu, kde odečítám 1, pokud je rozdíl větší než -20Hz případně pokud je menší, posílám hodnotu opět přímo na výstup.

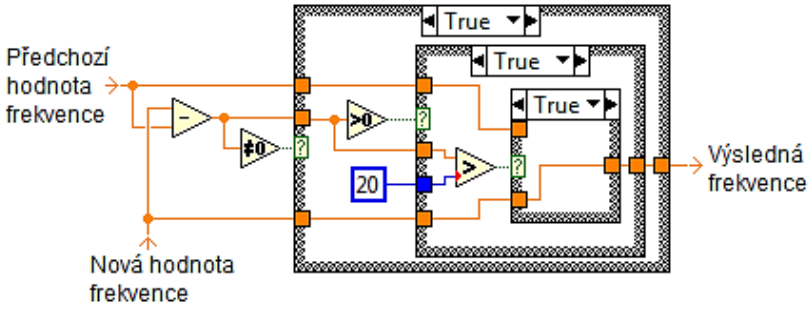
Blokové schéma filtru a sekvenci podmínek ukazuje obrázek č.14, praktickou realizaci kódu při různé kombinaci podmínek ukazují obrázky č.15, č.16 a č.17. Ukázka kódu v příloze na CD se nachází v adresáři Programy, uvedený kód v LabVIEW je uložen pod názvem **PLL using CO filtrace kmitání měřené frekvence.vi** .



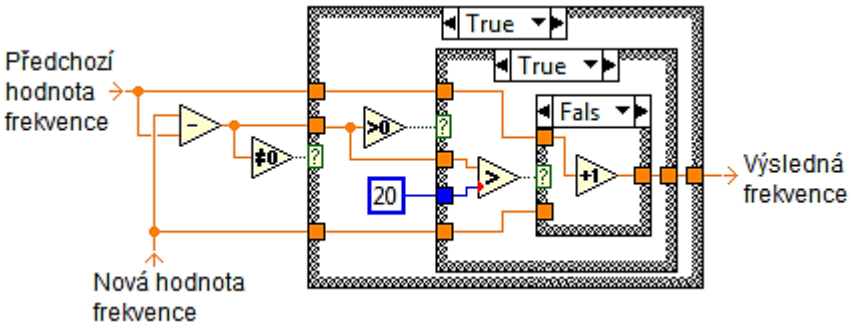
Obr. 14: Blokové schéma filtru zpomalení změny



Obr. 15: Přenos změřené frekvence při 0 rozdílu vůči předchozí



Obr. 16: Přenos změřené frekvence při rozdílu > 20Hz oproti předchozí



Obr. 17: Přenos změřené frekvence při rozdílu < 20Hz oproti předchozí

Filtrací měřené frekvence se chování programu znatelně zlepšilo a hodnota měřené frekvence se téměř úplně ustálila, na druhou stranu zhoršila se odezva na změnu frekvence měřeného signálu díky vlastnostem filtru.

Po mnoha různých testech a pokusech jsem nakonec přišel na původ problému se zvlněním frekvence měřeného signálu. Naměřená data získávám z měřící karty jako datový typ Waveform. Tento datový typ představuje skupinu informací o době počátku měření (t0), rychlosti vzorkování (dt), poli naměřených hodnot (Y) a vlastnostech měřeného signálu (název kanálu, typ měřené veličiny, jednotky, ...) (attributes). Problémem je, že měřící karta a smyčka měření analogového vstupu neumí změnit informaci o vzorkovací frekvenci (dt – doba mezi 2 vzorky (s)) za běhu měření. Pamatuje si pouze hodnotu nastavenou při inicializaci AI. Vyčtené Tw tedy nese informaci o vzorkování s rychlostí 1kHz bez ohledu na skutečnou rychlost vzorkování a následná analýza frekvence signálu způsobuje nežádoucí kmitání.

3.2.3 Oprava informace o vzorkování

3.2.3.1 Rozbor problému

Mějme sinusový měřený signál o frekvenci f_{sig} = 55Hz, periodě T_{sig} = 0,01818s, počet vzorků v Tw je konstantní n = 200 a vzorkovací frekvenci při inicializaci f_{VZ} = 1kHz a periodě T_{VZ} = 0,001s. Při prvním průchodu cyklem měření měřící karta navzorkuje signál zadanou rychlostí (1kHz), vyčte 200 naměřených hodnot, tj. doba trvání časového okna T_{TW} (s) představuje:

$$n \cdot T_{VZ} = T_{TW}\ (s) \qquad 200 \cdot \frac{1}{1000} = 0{,}2s \tag{7.}$$

Program provede analýzu frekvence signálu dle počtu period a dle informace o vzorkovací frekvenci a vypočte frekvenci signálu (T_{sig} – perioda měřeného signálu, f_{sig} frekvence měřeného signálu; T_{VZ} perioda vzorkování, f_{VZ} frekvence vzorkování; VZ/per počet vzorků na 1 periodu měřeného signálu).

$$\frac{T_{sig}}{T_{VZ}} = VZ/per \qquad \frac{0{,}01818}{0{,}001} = 18{,}18 \tag{8.}$$

$$\frac{1}{VZ/per \cdot T_{VZ}} = f_{sig} \qquad \frac{1}{18{,}18 \cdot 0{,}001} = 55Hz \tag{9.}$$

Protože vzorkovací frekvence odpovídá skutečné, program zjistí hodnotu 55Hz. Protože vzorkovací frekvence je 20 násobkem měřené frekvence, změní program vzorkování na 1100Hz. Při druhém průchodu cyklem opět vyčítáme 200 hodnot stále stejného signálu, jenže s vyšší vzorkovací frekvencí, aktuálně tedy doba trvání Tw činí:

$$n \cdot T_{VZ} = T_{TW}\ (s) \qquad 200 \cdot \frac{1}{1100} = 0{,}1818s \tag{10.}$$

Tímto způsobem obdržíme méně period měřeného signálu než v předchozím cyklu. Protože skutečná perioda vzorkování je menší, ale zpětný výpočet předpokládá stále stejnou frekvenci vzorkování (1kHz), budou analyzované hodnoty následující:

$$\frac{T_{sig}}{T_{VZ}} = VZ/per \qquad \frac{0{,}01818}{0{,}000909} = 20 \tag{11.}$$

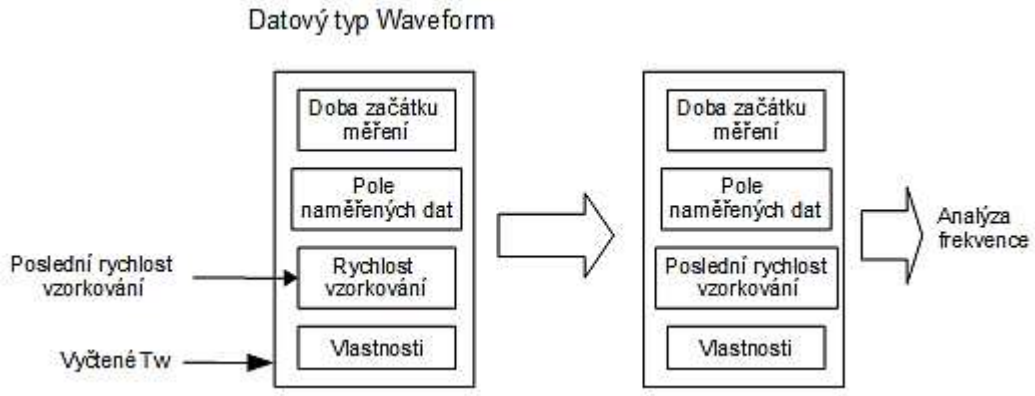
$$\frac{1}{VZ/per \cdot T_{VZ}} = f_{sig} \qquad \frac{1}{20 \cdot 0{,}001} = 50Hz \tag{12.}$$

Počet vzorků na periodu měřeného signálu vzroste, analýza frekvence detekuje 50Hz. Program nastaví vzorkování na 20 násobek změřené frekvence, tj. 1kHz a při dalším průchodu cyklem se děj opakuje. Program tedy kmitá okolo skutečné frekvence signálu a samostatně není schopen detekovat skutečnou hodnotu frekvence měřeného signálu. Při předchozích pokusech o nápravu filtrací programy

prováděly filtraci na hodnotu blízkou střední hodnotě měřených frekvencí. Účinnější nápravou této chyby je provést nahrazení nekorektní informace o vzorkování správnou hodnotou.

3.2.3.2 Řešení problému

Protože blok AI měřící karty za běhu neukládá informace o změně vzorkování, nýbrž s vyčítaným Tw ve formátu datového typu Waveform posílá informace o hodnotě nastavené při inicializaci, vytvořil jsem v programu proměnnou, která na konci každého cyklu uloží aktuální vzorkovací frekvenci a na začátku následujícího cyklu přepíše nesprávnou hodnotu ve vyčteném Tw, obrázek č. 18. Praktickou realizaci této úpravy ukazuje program v příloze na CD v adresáři Programy, kód v LabVIEW je pojmenován **PLL using CO odstranění kmitání měřené frekvence.vi** .



Obr. 18: Oprava vzorkovací frekvence

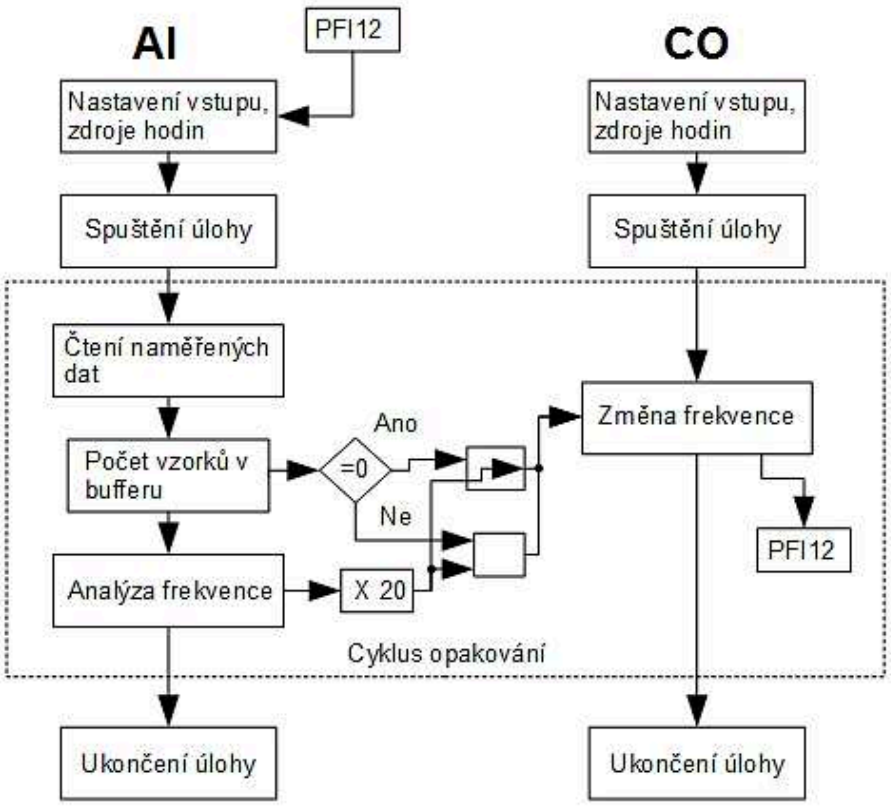
Nyní tedy provádím analýzu měřeného signálu se správnou informací o rychlosti vzorkování a již nedochází ke kmitání měřené frekvence. Nyní je také možno odstranit filtry, čímž se zlepší odezva na změnu měřeného signálu a zrychlí zavěšování PLL. Problémy nastávají při vytížení procesoru, kdy dochází ke zpomalení regulační smyčky a opět ke kmitání frekvence měřeného signálu.

3.3 Kontrola stability PLL při vytížení procesoru (zpomalení řetězce regulace)

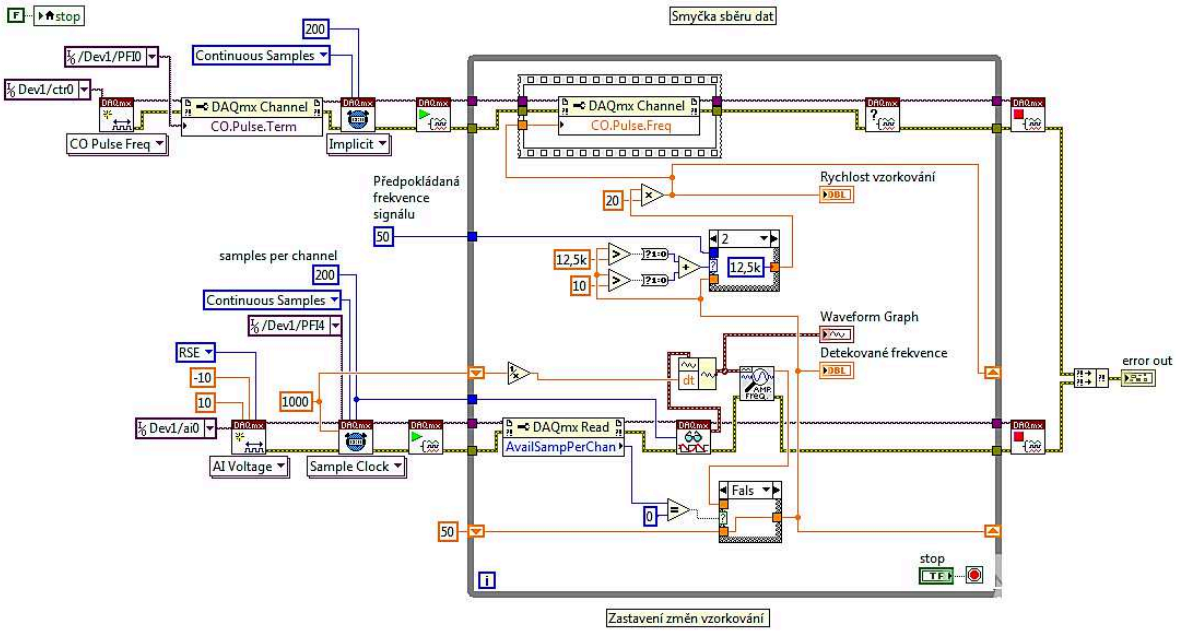
Provedl jsem testování programu při vytížení systému a sledoval jeho chování. Vytížení procesoru jsem realizoval programem provádějícím výpočty frekvenčních spekter, RMS a střední hodnoty signálu a dalšími programy jako antivirová kontrola PC a stahování dat z internetu. Při vytížení dochází ke zpomalení regulační smyčky a zároveň ke zpomalení vyčítání dat z měřící karty. Protože výrobce tento stav předpokládal, obsahuje měřící karta zásobník (buffer) pro ukládání naměřených dat, která programem vyčítáme a uvolňujeme tak místo dalším. Množství uložených dat je omezené velikostí zásobníku, a při delší době nevyčítání dojde k přepisu naměřených hodnot novými daty. Při zpoždění programu se nevyčtená data začnou shromažďovat v zásobníku a vzniká nebezpečí změny rychlosti vzorkování v průběhu měření Tw. Takto naměřené Tw obsahuje signál s neekvidistantním vzorkováním a analýza frekvence takového signálu je zatížena velkou chybou. Dalším nebezpečím je, že analyzovaný signál již není aktuální, pochází z času více či méně vzdáleného okamžiku a frekvence signálu může být značně odlišná oproti právě vzorkované. Při pokračování ve změnách vzorkování začne narůstat chyba celistvosti počtu period v měřeném Tw a program přestává plnit funkci PLL.

3.3.1 Udržení stability regulátoru zastavením změn vzorkování

Jedním z možných řešení tohoto problému je kontrolovat počet nevyčtených dat v zásobníku pomocí vlastnosti uzlu DAQmx Read (DAQmx Read Property Node Available Samples Per Chanel) a při nárůstu zastavit změny vzorkovací frekvence, dokud nedojde k opětovnému uvolnění a vyčtení všech dat ze zásobníku. Blokové schéma programu znázorňuje obrázek č. 19, ukázku kódu znázorňuje obrázek č. 20. Ukázka programu je umístěna v příloze na CD v adresáři Programy, kód v LabVIEW je uložen pod názvem **PLL using CO zastavení změn vzorkování.vi** .



Obr. 19: Blokové schéma zastavení změn vzorkování



Obr. 20 Ukázka kódu zastavení změn vzorkování

Tímto řešením odstraním problémy při zpoždění regulační smyčky, ale zastavením změn vzorkování přestanu udržovat celistvý počet period měřeného signálu v Tw a program přestane plnit svou funkci.

3.3.2 Udržení stability regulátoru změnou počtu vzorků Tw

Dalším možným řešením problému stability při vytížení regulační smyčky je změnit počet vyčítaných vzorků a tím zajistit rychlé vyprázdnění zásobníku. Problém nastává v tom, že počet vyčítaných vzorků nelze měnit libovolně. Pravidlo pro řízení počtu vyčítaných vzorků vychází z níže popsaných rovnic kde: f_{vz} představuje vzorkovací frekvenci, a násobek (konstantu), f_{sig} frekvenci měřeného signálu, n počet vyčítaných vzorků. Závislost délky Tw (s) na počtu vzorků a vzorkovací frekvenci (která je $a = 20$ násobkem měřeného signálu viz rovnice č. 13) ukazuje rovnice č. 14. Poměr délky vyčítaného Tw a periody měřeného signálu udává počet period měřeného signálu v Tw viz rovnice č. 16.

$$f_{vz} = a \cdot f_{sig} \tag{13.}$$

$$n \cdot \frac{1}{f_{vz}} = n \cdot \frac{1}{a \cdot f_{sig}} = Tw(s) \tag{14.}$$

$$\frac{1}{f_{sig}} = perioda(s) \tag{15.}$$

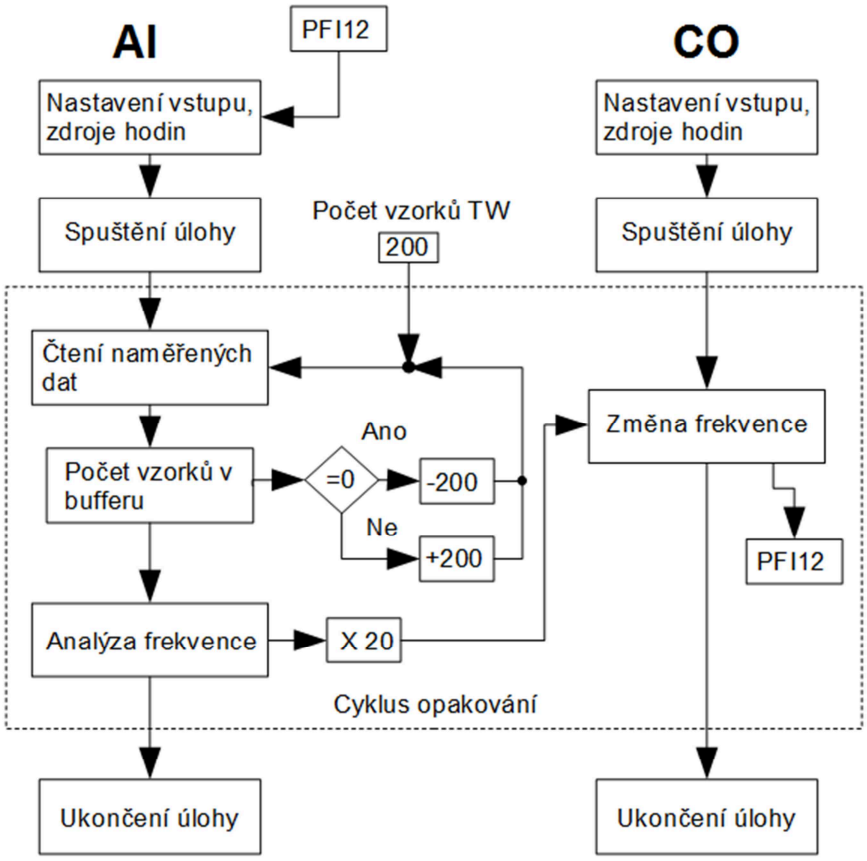
$$\frac{n \cdot \frac{1}{a \cdot f_{sig}}}{\frac{1}{f_{sig}}} = počet\ period\ v\ Tw \tag{16.}$$

$$\frac{n}{a} = počet\ period\ v\ Tw \tag{17.}$$

Úpravou jsem zjistil, že celistvost period měřeného signálu (rovnice č. 17) je závislá na poměru počtu vyčítaných vzorků a podílu vzorkovací frekvence a frekvence signálu (*a)* (rovnice č. 13). Protože hodnotu násobku neměním (je konstantní) a její hodnota je 20, připadá v úvahu měnit počet vyčítaných vzorků pouze v násobcích 20. Při řešení programu jsem zvolil vyčítání 10 period měřeného signálu a jejich násobky, takže:

$$n = a \cdot počet\ period\ v\ Tw \qquad n = 20 \cdot 10 = 200\ vzorků \tag{18.}$$

Praktickou realizaci této části programu představuje blokové schéma řízení počtu vzorků obrázek č.21.



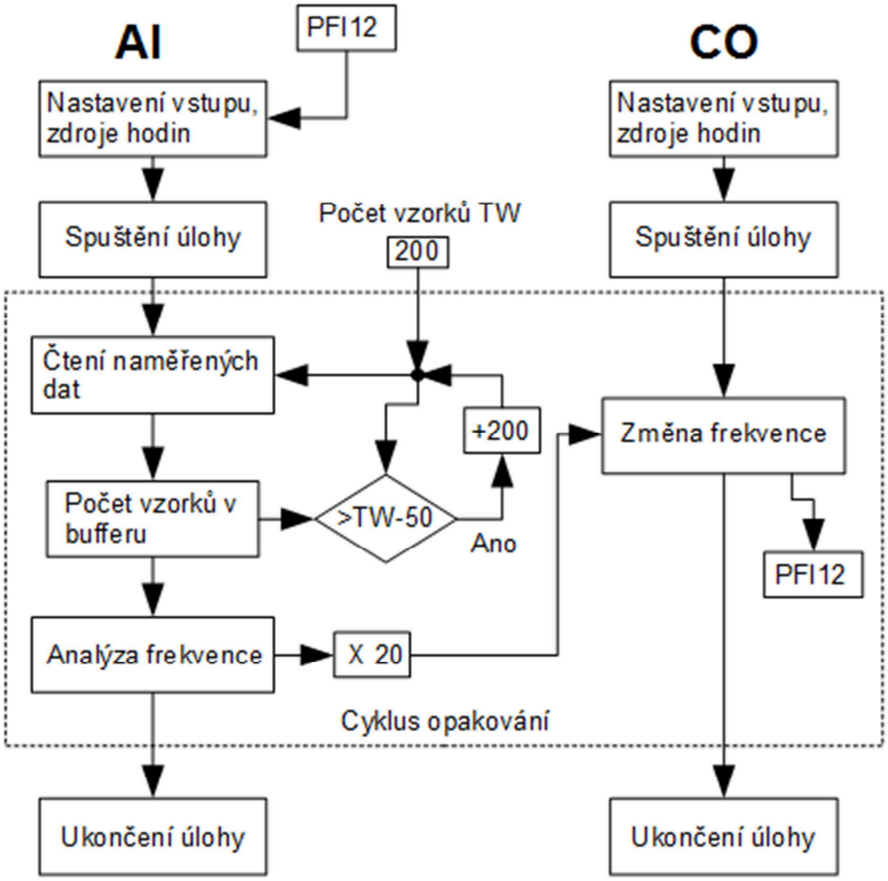
Obr. 21: Blokové schéma změny počtu vzorků

V programu provádím vyčítání násobků 200 vzorků. Tímto způsobem lze zajistit funkčnost regulátoru jak při zpoždění vlivem vytížení systému, tak i pro vyšší frekvence, kde lze vhodnou volbou počtu vzorků a vzorkovací frekvence docílit téměř libovolnou dobu měření Tw. Pro řízení počtu vzorků je ale nutné zajistit přesné a jednoznačné podmínky, jinak program nestíhá zpracovávat data rychle přicházející v malém počtu nebo zbytečně dlouho čeká na naměření velkého množství dat.

3.3.3 Podmínky pro změnu počtu vzorků v Tw

Protože je důležité udržet konstantní vzorkovací frekvenci v celém Tw, řídím počet vyčítaných vzorků primárně dle stavu zásobníku měřící karty a to tak, aby při změně vzorkování nezůstávaly v zásobníku nevyčtené vzorky. V měřícím řetězci AI lze pomocí uzlu vlastností DAQmx Read (DAQmx Read Available Samples Per Chanel) vyčíst aktuální množství vzorků v zásobníku měřící karty. Toto čtení provádím před samotným vyčítáním naměřených dat. Protože počítám s malou časovou prodlevou u funkce vyčítání dat, hlídám, aby v zásobníku bylo o 50 vzorků méně, než je celkový počet vyčítaných vzorků. Pokud bude počet vzorků vyšší a hrozí, že by nedošlo k vyčtení všech dat zásobníku, zvýším počet vyčítaných vzorků a tím urychlím vyprázdnění zásobníku. Blokové schéma kódu ukazuje obrázek č. 22, ukázka programu je v příloze na CD

v adresáři Programy, kód v LabVIEW je uložen pod jménem **PLL using CO řízení počtu vzorků Tw 1.vi** .



Obr. 22: Blokové schéma zajištění konstantní vzorkovací frekvence v celém Tw

Tímto způsobem lze zvyšovat počet vyčítaných vzorků v závislosti na zpoždění regulační smyčky (čím větší zpoždění, tím více nevyčtených vzorků v zásobníku). Nevýhodou je, že optimální a nejčastější stav zásobníku je 0, a tedy nelze podle jednoznačné podmínky počet vzorků snižovat a v důsledku toho po chvilkovém vytížení systému, kdy počet vzorků v zásobníku narostl, bude program zbytečně čekat na větší množství vzorků, než je potřeba.

3.3.3.1 Doplnění podmínek pro změnu počtu vzorků Tw

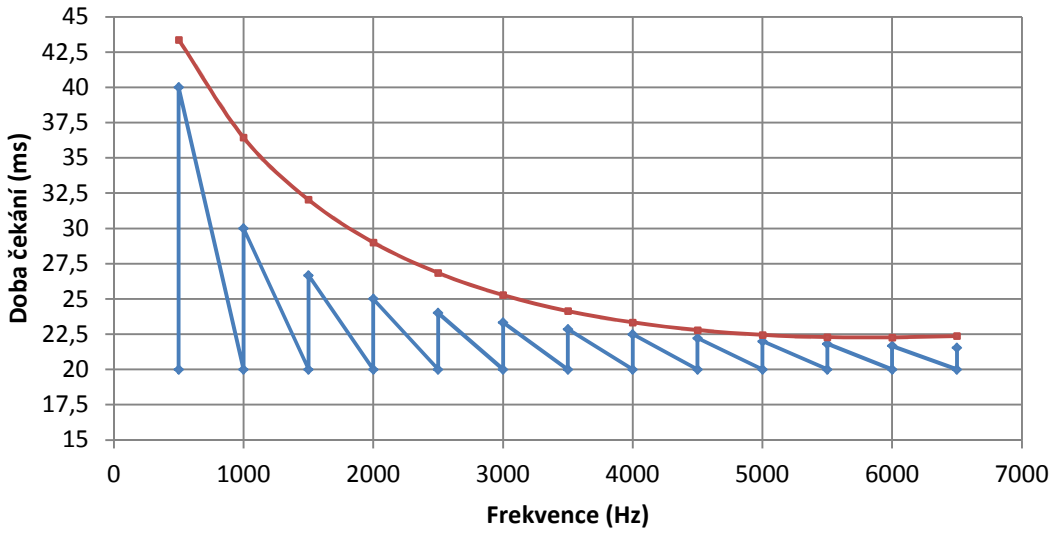
Pro doplnění podmínek, aby nedocházelo ke stavu, kdy program zbytečně očekává příliš velké množství vzorků, jsem přišel s nápadem měřit dobu od času zjištění počtu vzorků v zásobníku po čas obdržení naměřených dat. V normálním stavu, kdy procesor není vytížen a program běží rychle, tato doba představuje čas potřebný ke změření žádaného počtu vzorků (doba Tw (s)). V případě nárůstu

frekvence měřeného signálu nebo krátkého zpoždění programu PLL dojde ke zkrácení doby potřebné k vyčtení Tw. Pokud se tato doba zkrátí výrazně, ale zároveň nebude splněna předchozí podmínka (počet vzorků v zásobníku nepřesáhne maximum – 50), zvýším počet vyčítaných vzorků a snažím se tímto způsobem udržet dobu čekání nad hodnotou 20ms. Tuto hodnotu horní meze čekání (20ms) jsem nastavil záměrně, a to aby docházelo ke změnám počtu vzorků při známých frekvencích měřeného signálu. Výpočet frekvence, při které dojde k přepnutí na vyšší počet vzorků, ukazuje rovnice č. 19.

$$Tw\,(s) = n \cdot T_{VZ} = \frac{n}{f_{VZ}} = \frac{n}{20 \cdot f_{sig}} \quad \gg \quad f_{sig} = \frac{n}{20 \cdot Tw} \tag{19.}$$

$$f_{sig} = \frac{200}{20 \cdot 0{,}02} = 500Hz \tag{20.}$$

Dosazením (rovnice č. 20) jsem zjistil frekvenci přepnutí z 200 na 400 vzorků pro signál o frekvenci 500Hz. Dosazené výsledky dalších hodnot ukazuje tabulka Tab1. Ke zvýšení počtu vyčítaných vzorků dojde při zvýšení frekvence signálu o každých 500Hz (řízení pomocí spodní meze čekání). Grafické vyjádření předpokládaného průběhu doby čekání na vyčtení Tw ukazuje modrá křivka v grafu č. 7.



Graf 7. Předpokládaný průběh doby čekání na data v závislosti na frekvenci měřeného signálu

Pro ubírání počtu vzorků (řízení pomocí maximální doby čekání (horní meze čekání)) je nutné aproximovat spojnici špiček křivky řízení pomocí spodní meze čekání (modrá křivka) v grafu č. 7. Aproximaci jsem provedl v MS Excel, jako závislost doby čekání na vyčtení Tw (s) na počtu vyčítaných vzorků n. Tuto závislost přepočtenou na frekvenci měřeného signálu ukazuje červená křivka v grafu č. 7 a popisuje rovnice č. 21.

$$Tw\ (s) = n^{-0{,}0245} + n \cdot 8 \cdot 10^{-6} - 0{,}8234 \qquad (21.)$$

$$Tw\ (s) = 400^{-0{,}0245} + 400 \cdot 8 \cdot 10^{-6} - 0{,}8234 = 0{,}04327 \qquad (22.)$$

Dosazením (viz rovnice č. 22) obdržím hodnoty frekvence měřeného signálu, kdy dojde ke změně počtu vzorků (ke snížení počtu vzorků). Vypočtené hodnoty ukazuje tabulka Tab2.

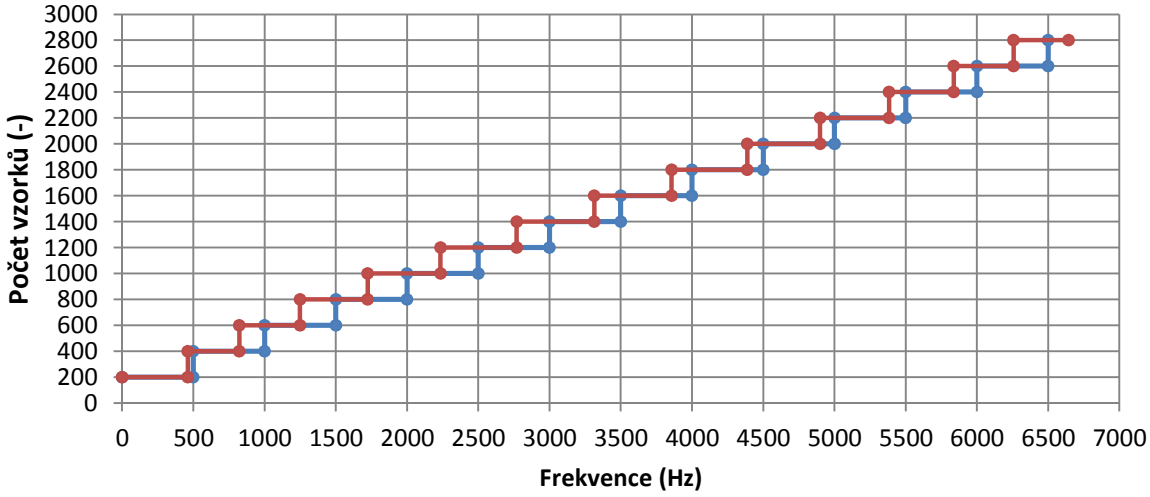
Tab. 1 Rostoucí frekvence signálu

Rostoucí frekvence signálu		
Počet vzorků (n)	Doba čekání T (ms)	Frekvence f_{sig} (Hz)
200	0,02	500
400	0,02	1000
600	0,02	1500
800	0,02	2000
1000	0,02	2500
1200	0,02	3000
1400	0,02	3500
1600	0,02	4000
1800	0,02	4500
2000	0,02	5000
2200	0,02	5500
2400	0,02	6000
2600	0,02	6500
2800	0,02	7000
3000	0,02	7500

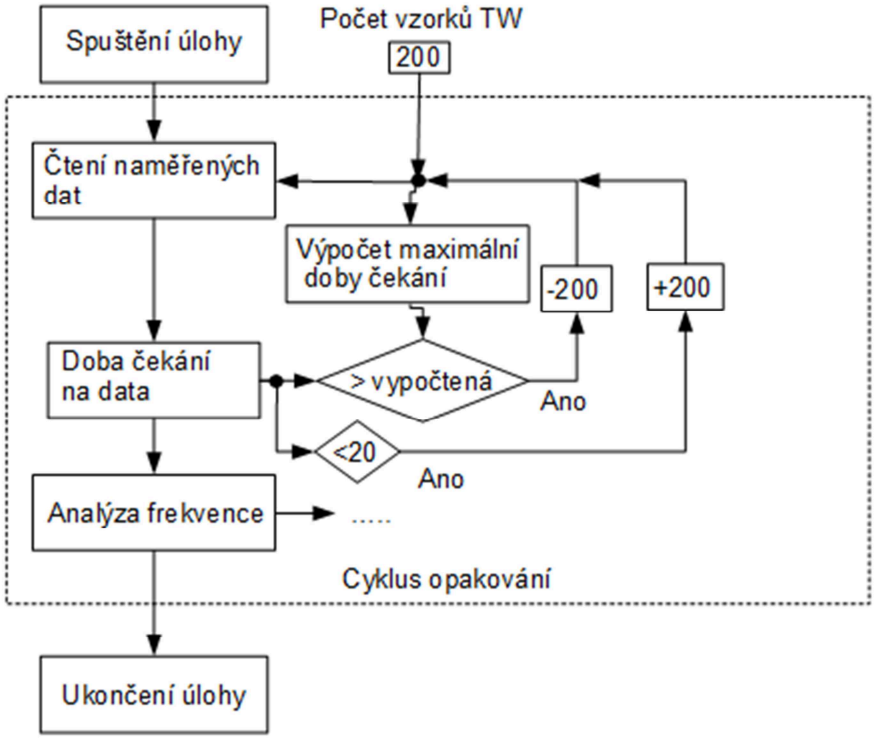
Tab. 2 Klesající frekvence signálu

Klesající frekvence signálu		
Počet vzorků (n)	Doba čekání T (ms)	Frekvence f_{sig} (Hz)
400	0,04327	462,1
600	0,03634	825,6
800	0,03194	1252,6
1000	0,02891	1729,7
1200	0,02674	2243,6
1400	0,02518	2780,6
1600	0,02404	3327,8
1800	0,02324	3873,5
2000	0,02269	4407,4
2200	0,02235	4921
2400	0,02219	5407,9
2600	0,02217	5863,6
2800	0,02227	6285,2
3000	0,02248	6671,4

Protože při nižších frekvencích měřeného signálu než 200Hz by program vyčítal 0 vzorků, kontroluji, zda počet vyčítaných vzorků není menší než 190, v případě že ano, nastavím počet vzorků zpět na hodnotu 200. Výsledný průběh řízení počtu vzorků dle frekvence měřeného signálu (dle délky doby potřebné k vyčtení Tw (s)) ukazuje graf č. 8. Modrá křivka představuje změny počtu vyčítaných vzorků při zvyšování frekvence měřeného signálu, červená křivka změny počtu vyčítaných vzorků při snižování frekvence měřeného signálu. Blokové schéma programu s řízením počtu vzorků ukazuje obrázek č. 23, ukázka kódu je umístěna v příloze na CD v adresáři Programy, kód v LabVIEW je uložen pod jménem **PLL using CO řízení počtu vzorků Tw 2.vi** .



Graf 8. Graf průběhu řízení počtu vzorků v závislosti na frekvenci signálu



Obr. 23: Blokové schéma řízení počtu vzorků

Tímto způsobem jsem schopen měřit signály mnohem vyšších frekvencí než předpokládaných 50Hz a zachovat funkci programu i při zpoždění řídící smyčky vlivem vytížení procesoru. Drobným problémem je, že ke změnám vytížení procesoru dochází skokově, a tedy doba čekání na vyčtení Tw se taky mění skokově a v programu by takovýto průběh řídící veličiny způsoboval kmitání počtu

vyčítaných vzorků. Aby bylo možno použít takto získané hodnoty k řízení programu, využil jsem filtru výpočtu průměrné hodnoty popsaného v kapitole 3.2.1 a tím stabilizoval tyto kmity.

3.4 Ukládání proměnných za běhu programu

Abych mohl zpětně zjišťovat reakce programu na měřený signál, provádím ukládání aktuálních hodnot proměnných v programu. V průběhu každého cyklu programu shromažďuji tato data ve skupině dat (cluster), na konci cyklu řídící smyčky odešlu skupinu pro zpracování do prostřední smyčky, kde provedu vykreslení do grafu typu Waveform Chart a grafu typu Waveform Graph a data určená k uložení přeformátuju do pole hodnot. Zároveň, protože v hlavní smyčce (řídící smyčka) vyčítám proměnný počet vzorků, provádím v prostřední smyčce rozdělení naměřených dat na Tw po 200 vzorcích. Pole hodnot pomocí fronty přeposílám do třetí samostatné smyčky programu (smyčka ukládání), kde provedu uložení do binárního souboru.

3.5 Výsledné řešení programu

Ze získaných poznatků jsem sestavil komplexní řešení programu, na kterém budu následně provádět testování. Program se skládá ze 3 bloků (3 smyček while).

- Hlavní regulační smyčka PLL – zde funguje řízení vzorkovací frekvence, řízení počtu vyčítaných vzorků a samotné vyčítání naměřených dat.

- Prostřední smyčka – obstarává úpravu měřených dat pro následné další využití (udržuje počet vzorků v Tw na hodnotě 200), provádí vykreslování hodnot z proměnných hlavní smyčky do grafů a provádí formátování dat pro následné uložení do souboru.

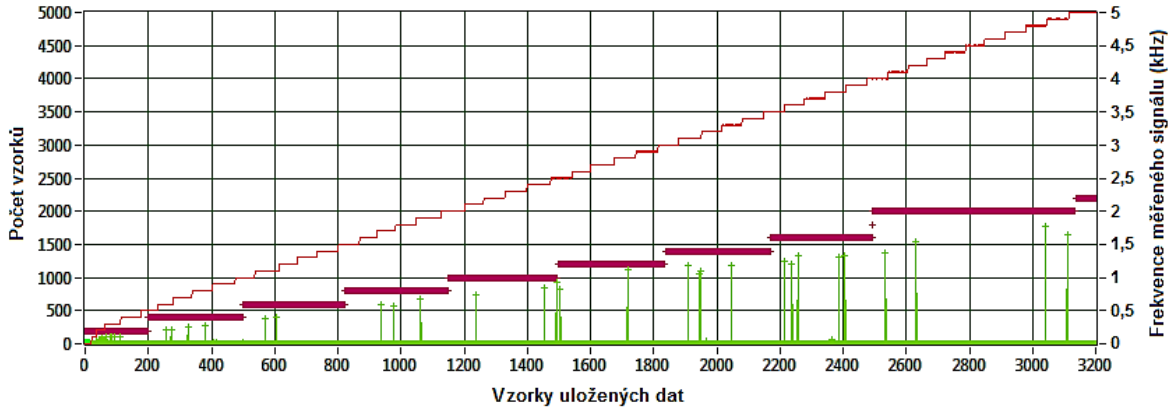
- Smyčka ukládání - zajišťuje uložení naformátovaných dat do binárního souboru pro pozdější analýzu.

Vstupním parametrem programu jsou tedy naměřená data přivedeného analogového periodického signálu s proměnnou frekvencí a výstupem data ve formátu Waveform s celistvým počtem period a 200 vzorky měřeného signálu. Blokové schéma programu viz příloha [Blokové schéma výsledného programu], ukázka programu viz příloha na CD v adresáři Programy, kód v LabVIEW je uložen pod jménem **PLL using CO výsledné řešení.vi** .

4 Testování programu

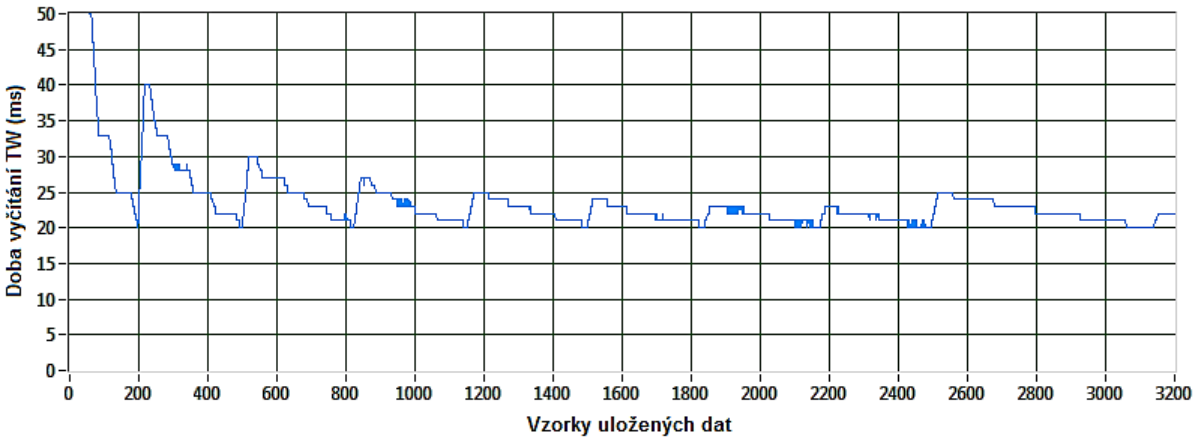
4.1 Průběh řízení počtu vyčítaných vzorků

Jednou z ukládaných proměnných je doba čtení dat. Představuje časovou prodlevu od zjištění počtu vzorků v zásobníku do fyzického získání měřených dat (doba měření Tw). S rostoucí frekvencí měřeného signálu tato doba klesá a v závislosti na jejích aktuálních hodnotách řídím počet vyčítaných vzorků. Z uložených dat lze tedy vyčíst průběh doby měření Tw (obrázek č. 33) a průběh regulace počtu vzorků a průběh frekvence měřeného signálu viz graf č. 9. Křivky v grafech představují jednotlivé snímané hodnoty. Červená křivka představuje frekvenci měřeného signálu (kHz), fialová křivka počet vyčítaných vzorků a zelená křivka počet vzorků v zásobníku.



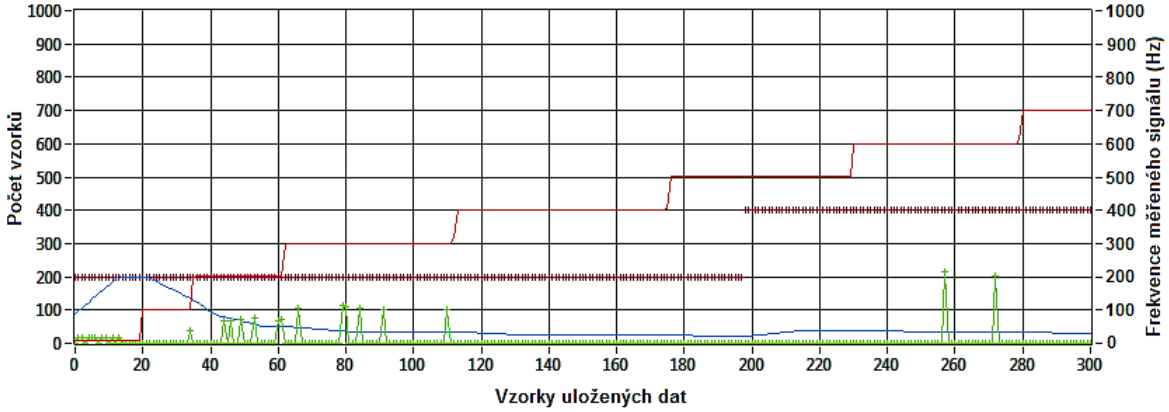
Graf 9. Graf průběhu řízení Tw

Zaostřením grafu č. 9 dostaneme skutečný průběh křivky doby čekání na vyčtení Tw (modrá křivka) viz graf č. 10, který lze porovnat s předpokládaným průběhem doby čekání v grafu č. 7.



Graf 10. Graf průběhu doby čekání na data (ms)

Zaostřením grafu č. 9 na začátek měřeného úseku získáme průběh řízení v oblasti nízkých frekvencí.



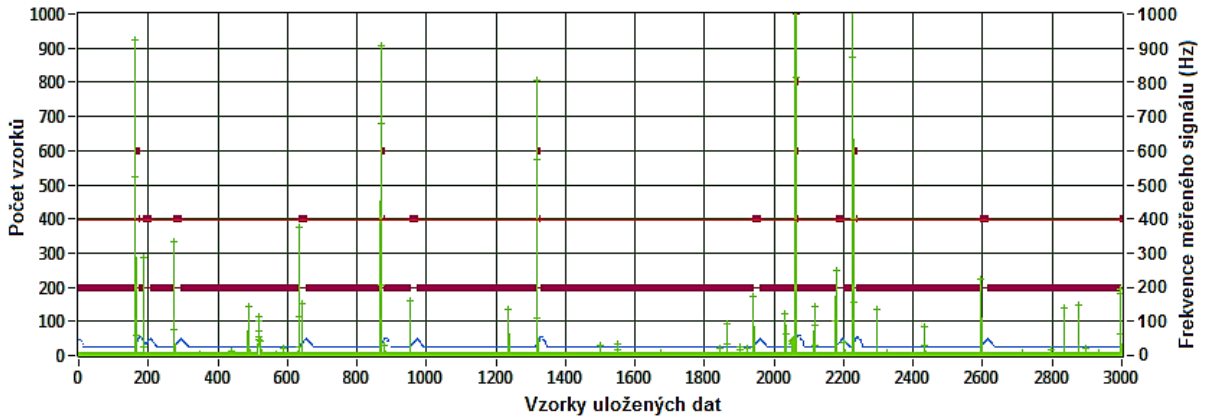
Graf 11. Graf průběhu řízení Tw oblast nízkých frekvencí

Porovnáním grafů průběhu doby čekání na naměřená data (graf č. 10) a předpokládaný průběh doby čekání na naměřená data (graf č. 7) jsem zjistil, že se předpokládaná křivka shoduje s naměřenou a k přepínání počtu vyčítaných vzorků dochází dle předpokladu, tj. co 500Hz. Z grafu průběhu měření v oblasti nízkých frekvencí (graf č. 11) můžeme pozorovat chování regulace PLL v oblasti frekvence předpokládaného signálu tj. 50Hz při normálním provozu (bez vytížení procesoru). Počet vyčítaných vzorků je na minimální hodnotě 200vz. a počet vzorků v zásobníku se pohybuje okolo hodnoty 0 vzorků.

4.2 Průběh řízení Tw v závislosti na vytížení systému

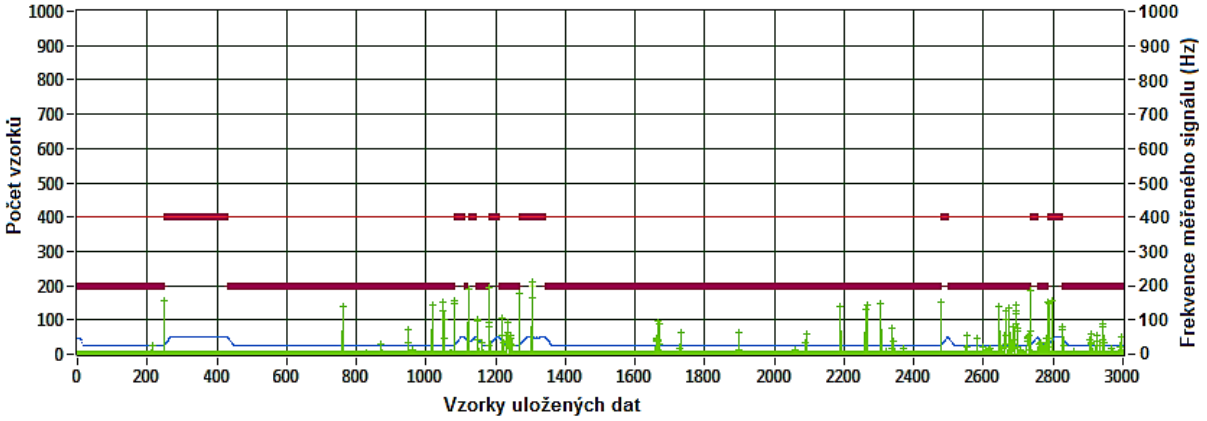
Cílem testu je zjistit chování programu v závislosti na vytížení procesoru systému, na kterém běží. Testovaný HW je notebook s procesorem Intel I5 2.4GHz s možností přetaktování na 3.0GHz, 4GB ram, SSD HDD. Měřený signál je sinusový signál s konstantní frekvencí 400Hz.

Provedl jsem testování vlastností regulační smyčky bez vytížení PC, na kterém program běží. Protože program nastavuje automaticky parametry řídící smyčky tak, aby zvládl zpracovávat data s předem danou odezvou, viz kapitola 3.3.3. Předpokládal jsem, že ke zpožděním nebude docházet vůbec, případně že budou zpoždění krátká, způsobená vlivem přepínání úloh běžících na procesoru testovaného PC. Průběh naměřených dat ukazuje graf č. 12. Červená křivka představuje frekvenci měřeného signálu (Hz), fialová křivka počet vyčítaných vzorků, zelená křivka počet vzorků v zásobníku a modrá křivka dobu čekání na vyčtení Tw.



Graf 12. Graf hodnot bez vytížení systému

Druhou částí tohoto testu je testování vlastností regulační smyčky během vytížení procesoru systému díky náročným výpočtům s daty a programům náročným na čas procesoru. Tentokrát jsem předpokládal zpoždění s delší dobou trvání než v předchozím případě díky zpracovávání většího množství úloh. Pro simulaci zátěže systému jsem vytvořil program generující z náhodných hodnot signál, následně provádí měření amplitudového, výkonového frekvenčního spektra tohoto signálu, výpočet RMS a střední hodnoty signálu, provádí matematickou operaci násobení amplitudového spektra signálu a získané hodnoty vykresluje do grafů a indikátorů. Pro zajištění dostatečného vytížení systému jsem spustil stahování souborů z internetu a kontrolu počítače antivirem. Získané výsledky ukazuje graf č. 13. Červená křivka představuje frekvenci měřeného signálu (Hz), fialová křivka počet vyčítaných vzorků, zelená křivka počet vzorků v zásobníku a modrá křivka dobu čekání na vyčtení Tw.



Graf 13. Graf hodnot během vytížení systému

Porovnáním grafů č. 12 a č. 13 vidíme značný rozdíl v chování řídícího programu a to v počtu vzorků v zásobníku dat. Množství těchto vzorků představuje délku zpoždění před začátkem vyčítání naměřených dat. Příčinou zřejmě je, že během normálního běhu systému se procesor automaticky nastaví do režimu snížené spotřeby a nízkého výkonu a sníží nároky na dodržení přiděleného

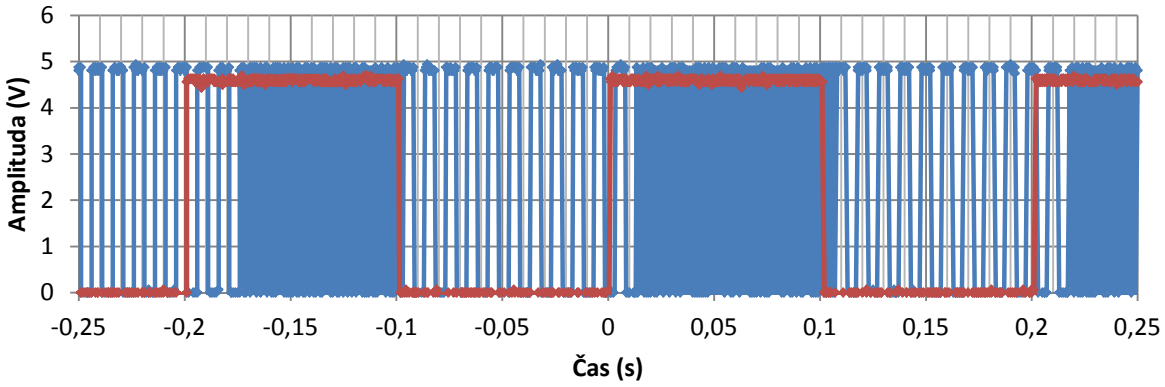
systémového času běžícím vláknům. Naopak během vytížení se pohybovala rychlost procesoru okolo maximální povolené frekvence (automatické řízení taktu procesoru dle vytížení). Také došlo ke zpřísnění nároků na vrácení přiděleného systémového času vlákny. Chování samotné řídící smyčky programu odpovídá předpokládanému chování, kdy program reaguje na množství dat v zásobníku a řídí počet vyčítaných vzorků dle doby čekání na naměřená data.

Pro optimální chod programu je vhodnější častější zpoždění s krátkou dobou trvání (graf č. 13), protože vhodnou volbou počtu vyčítaných vzorků program nastaví optimální dobu čekání na Tw, která díky ne příliš velkému množství dat v zásobníku není příliš dlouhá a po odeznění zpoždění rychle vrátí počet vyčítaných vzorků na původní hodnotu. Opačně tomu je v případě občasných zpoždění s dlouhou dobou trvání (graf č. 12), kde program nastavuje pro velké množství dat v zásobníku velký počet vyčítaných vzorků a po odeznění zpoždění trvá, než se počet vyčítaných vzorků postupně vrátí na původní hodnotu a dochází ke zbytečnému čekání.

4.3 Testování odezvy algoritmu na změnu vstupního signálu

Tímto testem jsem ověřil rychlost reakce algoritmu na změnu signálu na vstupu měřící karty. Vstupním signálem provádějícím změnu je obdélníkový signál nízké frekvence z laboratorního generátoru, výstupním signálem indikujícím změnu je signál s proměnnou frekvencí závislou na polaritě obdélníkového signálu. Oba tyto signály zobrazuji na osciloskopu.

Programem vytvořeným k účelu tohoto testu měřím hodnoty digitálního vstupu a časovačem generuji signál proměnné frekvence. Dle polarity měřeného signálu provádím výběr mezi generováním 100Hz signálu (pro logickou „0“) a generováním 1kHz signálu (pro logickou „1“). Zároveň měřím dobu trvání cyklu programu. Průběh naměřených hodnot ukazuje graf č. 14. Červená křivka představuje řídící signál z laboratorního generátoru, modrá křivka zobrazuje řízený signál generovaný měřící kartou.



Graf 14. Graf odezvy měřící karty

Odečtením hodnot z grafu jsem zjistit maximální dobu odezvy 25ms a nejkratší hodnotu 2ms. Odečítáním doby běhu cyklu z indikátoru programu jsem zjistil hodnotu 17ms. Porovnáním naměřených hodnot odhaduji, že doba odezvy na změnu vstupního signálu závisí nejen na okamžiku příchodu signálu (pozice v Tw) ale také na aktuálním zpoždění vlivem dočasného vytížení procesoru.

5 Závěr

Cílem této bakalářské práce je realizovat program fázového závěsu pro standardní kartu sběru dat a zhodnotit jeho vlastnosti.

Při realizaci programu jsem získal přehled o možnostech využití měřících karet a naučil se pracovat v programovacím prostředí LabVIEW.

Toto prostředí je grafické programovací prostředí pro tvorbu virtuálních měřících přístrojů. Vývoj programu sestává z propojování a parametrizace funkčních bloků do funkčního celku.

Cílem vyvíjeného programu je optimalizovat měření neznámého periodického signálu. Po seznámení s možnostmi měřící karty jsem se rozhodl jít cestou řízení celistvosti period měřeného signálu změnou vzorkovací frekvence. Provádím tedy výpočet frekvence přivedeného neznámého signálu a na základě této hodnoty nastavím optimální vzorkovací frekvenci.

V průběhu vývoje jsem zjistil, že největší problém těchto regulačních obvodů je problém s udržením stability programu. Ta může být narušena buď nestabilitou frekvence měřeného signálu, chybou v programu, dále vytížením procesoru systému nebo vysokou hodnotou frekvence měřeného signálu. Tyto nebezpečné stavy jsem programově ošetřil a testováním ověřil funkčnost. Ošetření jsem provedl tak, že měním počet vyčítaných vzorků, aby program stíhal zpracovávat naměřená data nebo zbytečně dlouze nečekal na jejich naměření.

Program lze aplikovat na všechny běžné měřící karty obsahující čítač a obvody měření analogového vstupu a rozšířit tak jejich možnosti využití na měření signálů napájecí sítě. Programem zajišťuji měření 10 period signálu proměnné frekvence od 10Hz nahoru.

Další vývoj programu bude probíhat v podobě implementace programu jako VI server. Touto úpravou dojde k oddělení měřící smyčky od zbytku programu (úprava dat, ukládání). Tím vznikne samostatný program DAQ, kterému mohu přiřadit vyšší prioritu, tím pádem zvýším rychlost odezvy PLL.

6 Literatura

[1.] National Instruments. *Getting Started with LabVIEW* [Brožura]. 8.20. Irsko, 2006.

[2.] Žídek Jan. Grafické *Programování ve vývojovém prostředí LabVIEW* [skriptum]. Ostrava, 2002, 215s.

[3.] Wittassek Tomáš. *Virtuální instrumentace 1* [skriptum]. Ostrava, 2012, 264s.

Přílohy

Příloha1: Blokové schéma výsledného programu

Příloha2: obsah přiloženého CD

1. Ukázky programů (adresář Programy)
2. Výsledky testování (adresář Testování)
3. Program pro zobrazení výsledků testování (adresář Testování, název Zobrazení dat)
4. Bakalářská práce ve formátu PDF